# Cisco Crosswork Situation Manager 7.2.x Developer Guide

Powered by Moogsoft AIOps 7.2

# Table of Contents

# Developer Guide

The Developer Guide provides resources for developers who want to perform advanced functions or build applications that integrate with Cisco Crosswork Situation Manager.

If you want to build a new integration or create a custom reporting dashboard, this guide outlines how you can expose API endpoints to invoke various actions and functionality.

You can use the following APIs and modules:

- Graze API: Main API that can call retrieve data, update data and call actions in Cisco Crosswork Situation Manager.
- Stats API: API that you can use to retrieve statistics from Cisco Crosswork Situation Manager for reporting and dashboards.
- Moobot Modules: You can create bots to perform automated tasks and expose functions in different Moolets.
- Programmatic LAM: You can use this custom polling LAM to accept API calls and parse the responses into Cisco Crosswork Situation Manager events.

You can also use these APIs to perform tasks such as Situation enrichment. See section 'Enrichment' in the *Cisco Crosswork Situation Manager Implementor Guide* for more information.

## Graze API

You can use the Graze API to perform actions including the following:

- Assign and de-assign alerts.
- Create and close Situations.
- Add processes and services.
- Create and update SAML realms.
- Create and delete maintenance windows.

The Graze API is useful if you want to perform repeated actions. For example, sending repeated cURL commands using a script. For example, you can create a ticketing integration to enable bi-directional communication between Cisco Crosswork Situation Manager and a ticketing system. Refer to the *Cisco Crosswork Situation Manager Integration Guide* for available ticketing integrations.

An integration can raise and close Cisco Crosswork Situation Manager alerts in line with ticketing events, assign users and replicate comments. For all available endpoints see Graze API.

## Stats API

You can use the Stats API to retrieve statistics about:

- Your Cisco Crosswork Situation Manager system's performance.
- Teams' performance.
- Individual users' performance.

You can use this statistical data to populate dashboards. For example, you can use call **`getNewEventsPerSituationsStats`** to see the noise reduction from events to Situations for your Cisco Crosswork Situation Manager system.

These endpoints have been designed and optimized for Grafana. See Stats API for all available endpoints. See the 'Grafana Setup Tutorial' in *Cisco Crosswork Integration Guide* for more installation and configuration instructions.

## Moobot Modules

You can create and configure Moobot modules to perform automated tasks and expose functions including:

- Access external databases.
- Access an external RESTful API via HTTP.
- Read configuration files within LAMbots and Moobots.
- Build a key value dictionary shared across Moobots.
- Query and manipulate entities in the Cisco Crosswork Situation Manager databases. See MoogDb V2 for all available methods.
- Send an email in response to events occurring in Cisco Crosswork Situation Manager.

For more information see Moobot Modules.

## Programmatic LAM

The Programmatic LAM is a custom polling LAM. It is an advanced version of the REST Client LAM. The REST Client LAM accepts a single API call and parses the responses it receives into Cisco Crosswork Situation Manager events. The Programmatic LAM can accept multiple calls but you must define the processing yourself in the LAMbot using JavaScript.

For more information see Programmatic LAM.

## Graze API

- Architecture
- Configure Tomcat
- API Definition
- Authentication Troubleshooting
- Endpoints
  - Alerts
  - Situations
  - User Management
  - Security Realms
  - Dashboards and Reporting
  - Workflow
  - POST Parameters
  - form-urlencoded
  - application/json
- HTTP Status and Error Codes

The Graze API acts as an integration point for external services and exposes selected Cisco Crosswork Situation Manager functionality to authorized external clients.

As you would with any API, use caution when employing the Graze API. Excessive requests can impact overall system performance. Take special care when using **`getSituationIds`** and **`getAlertIds`**. Overusing these endpoints can have a negative impact on the backend datastore.

To work with a Graze API expert on your solution, you can engage Cisco Technical Services. Otherwise contact Cisco Support.

## Architecture

The Graze API is implemented as a set of servlets running in the Cisco Crosswork Situation Manager Tomcat instance that handles external Graze requests, making the UI servlet calls directly via cross-contexts.

## Configure Tomcat

You must configure Tomcat to allow cross-context calls to be made by adding the following to the **`context.xml`** file in the Tomcat **`$APPSERVER_HOME/conf`** directory:

```
<Context crossContext="true">
```

## API Definition

All Graze requests use the following URL format, where **`<server>`** is the hostname of the machine running the UI :

```
https://<server>/graze/v1/<request_type>
```

For example:

```
https://localhost/graze/v1/authenticate
```

All requests (other than **`authenticate`**) require a basic authentication header or a valid **`auth_token`**. A valid **`authenticate`** request must be successfully made before any Graze API request is used without a basic authentication header.

Inactive sessions will be logged out after one hour, and a new **`authenticate`** request must be made to get a new valid **`auth_token`**.

Note:

If you are making regular Graze requests within a one hour timeframe, you are considered active and your session will not expire.

## Authentication Troubleshooting

If an error occurs during Graze login authentication, the following output is returned:

```
{"message":"User is not authenticated","statusCode":3001}
```

As a security precaution, no more specific information is returned. This prevents information being provided to potential attackers as to which part of the authentication failed (for example 'Password incorrect').

Entries in the log file **`catalina.out`** (at **`WARN`** level) provide more information on authentication errors:

1. The user is not assigned the Grazer role:

```
User [john] does not have graze permission
```

2. No user of that name:

```
User [NotAUser] account unknown in database
```

3. Incorrect password:

```
Password incorrect for user [graze]
```

# Endpoints

## Alerts

### addAlertCustomInfo

A POST request that adds and merges custom information for a specified alert.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | The alert ID. |
| **custom_info** | JSON Object | A JSON Object containing the custom information. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/addAlertCustomInfo" -H "Content-Type:
application/json; charset=UTF-8" -d '{"alert_id" : 9, "custom_info" : {
"field1" : "value1" , "field2" : "value2" , "field3" :
["item1","item2","item3"] , "field4" : {"field4-1" : "value4-1","field4-2"
: "value4-2"} }}'
```

Successful return:

NO RESPONSE TEXT

### assignAlert

A POST request that assigns the moderator to the alert for a specified alert ID and user ID.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | The alert ID. |
| **user_id** | Number | The user ID. |
| **username** | String | A valid username. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/assignAlert"
-H "Content-Type: application/json; charset=UTF-8" -d '{"alert_id" : 7,
"username" : "network1" }'
```

Successful return:

NO RESPONSE TEXT

### addAlertToSituation

A POST request that adds a specified alert to a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request |
| **alert_id** | Number | The alert ID |
| **sitn_id** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

curl Command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/addAlertToSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"alert_id" : 16, "sitn_id" : 7 }'
```

Successful Return:

NO RESPONSE TEXT

### assignAndAcknowledgeAlert

A POST request that assigns and acknowledges the moderator to the alert for a specified alert ID and user ID.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | The alert ID. |
| **user_id** | Number | The user ID. |
| **username** | String | A valid username. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

### closeAlert

A POST request that closes one or more alerts.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | A single alert ID. |
| **alert_ids** | Number list | A list of alert IDs. |
| **thread_entry_comment** | String | Optional comment. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/closeAlert"
-H "Content-Type: application/json; charset=UTF-8" -d '{"alert_ids" :
[78,234,737],"thread_entry_comment" : "Closing as agreed during team
discussion 1/1/2018" }'
```

Successful return:

NO RESPONSE TEXT

### deassignAlert

A POST request that de-assigns the current moderator from the alert for a specified alert ID.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | The alert ID. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/deassignAlert" -H "Content-Type:
application/json; charset=UTF-8" -d '{"alert_id" : 7}'
```

Successful return:

NO RESPONSE TEXT

### getAlertActions

A GET request that returns the actions for alerts, ordered most recent last. You can use the **from** and **to** arguments to specify a period that you want to retrieve alert actions for. If you do not specify these, actions for all dates and times are returned.

Request Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| **auth_token** | String | No | A valid **auth_token** returned from the **authenticate** request. |
| **alert_ids** | Number list | No | List of alert IDs. |
| **start** | Integer | Yes | Starting row from which data should be included. |
| **limit** | Integer | Yes | Maximum number of actions you want to return. |
| **actions** | Number list | No | List of action codes. If no action codes are specified, all action codes are returned. See Alert Action Codes for a list of action codes and their descriptions. Only action codes 8 (Alert Resolved) and 9 (Alert Closed) are valid. |
| **from** | Number | No | Start time (in Unix epoch time) of the period you want to retrieve alert actions for. |
| **to** | Number | No | End time (in Unix epoch time) of the period you want to retrieve alert actions for. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object which contains alert details from the following:

| Name | Type | Description |
|---|---|---|
| uid | Number | User ID. |
| action_code | Number list | Code for the action in the JSON object. See Alert Action Codes for a list of action codes and their descriptions. |
| description | String | Description of the action. |
| details | String | Details of the action. |
| type | String | Type of action. |
| alert_id | Integer | Alert ID. |
| timed_at | Integer | Time stamp of the action. |

Example

Example cURL command to return the first 50 actions for alert IDs 1, 2, 3, and 6 for action codes 8 (Alert Resolved) and 9 (Alert Closed):

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getAlertActions"
--data-urlencode 'alert_ids=[1, 2, 3, 6]' --data-urlencode 'actions=[8,
9]' --data-urlencode 'limit=50' --data-urlencode 'start=0'
```

Example cURL command to return the first 50 actions for action codes 8 (Alert Resolved) and 9 (Alert Closed) between Unix epoch times 1553861746 and 1553872546:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getAlertActions"
--data-urlencode 'actions=[8, 9]' --data-urlencode 'limit=50' --data-
urlencode 'start=0' --data-urlencode 'from=1553861746' --data-urlencode
'to=1553872546'
```

Successful return request:

```
[{
   "uid": 49,
   "action_code": 8,
   "description": "Alert Resolved",
   "details": {},
   "alert_id": 1,
   "timed_at": 1557504393
   }, {
   "uid": 49,
   "action_code": 9,
   "description": "Alert Closed",
   "details": {},
   "alert_id": 1,
   "timed_at": 1557504912
   }
}]
```


### getAlertDetails

A GET request that returns details, such as Description or Severity, of a specified alert.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | The alert ID. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object which contains alert details from the following:

| Name | Type | Description |
|---|---|---|
| **active_sitn_list** | Number list | A list of Situation IDs of the active Situations to which this alert belongs. |
| **agent** | String | The agent name associated with this alert. * |
| **agent_location** | String | The agent location associated with this alert. * |
| **alert_id** | Number | The alert ID. |
| **class** | String | The class associated with this alert. * |
| **count** | Number | The number of times that this alert has occurred. |
| **custom_info** | JSON Object | A JSON Object containing the custom information. |
| **description** | String | The description associated with this alert. * |
| **entropy** | Number | The entropy value of the alert, the measure of probability that an alert will arrive in the system at any given time.<br><br>This is a value between 0 (very certain) and 1 (very uncertain). |
| **external_id** | String | The external ID associated with this alert. * |
| **first_event_time** | Number | The timestamp (in Unix epoch time) of the first occurrence of this alert. |

| | | |
|---|---|---|
| **int_last_event_time** | Number | The internal Cisco Crosswork Situation Manager timestamp (in Unix epoch time) of the last occurrence of this alert. |
| **last_event_time** | Number | The timestamp (in Unix epoch time) of the last occurrence of this alert. |
| **last_state_change** | Number | The timestamp (in Unix epoch time) of the last status change of this alert. |
| **manager** | String | The manager name associated with this alert. * |
| **owner** | Number | The User ID of the user that this alert is assigned to. |
| **severity** | Number | The alert's severity as an integer:<br>0 Clear<br>1 Indeterminate<br>2 Warning<br>3 Minor<br>4 Major<br>5 Critical |
| **signature** | String | The unique alert identifier. |
| **significance** | Number | The alert's significance as an integer:<br>0 Collateral<br>1 Related<br>2 Impacting<br>3 Causal |
| **source** | String | The source associated with this alert. * |
| **source_id** | String | The source ID associated with this alert. * |
| **state** | Number | Indicates the lifecycle state of the alert. |
| **type** | String | The type associated with this alert. * |

* = These details are derived from the input event text field, via the LAMs.

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getAlertDetails"
--data-urlencode "alert_id=3968"
```

Successful request return:


```
{
   "active_sitn_list":[1],
   "agent":"TestBed",
   "agent_location":"localhost",
   "alert_id":10,
   "class":"WebMon",
   "count":2,
   "custom_info":null,
   "description":"Web Server HTTPD is DOWN",
   "external_id":"12345",
```

```
    "first_event_time":1416307126,
    "int_last_event_time":1416307188,
    "last_event_time":1416307131,
    "last_state_change":1416307144,
    "manager":"WebMon",
    "owner":2, "severity":0,
    "signature":"SIG:Web Server Down Trap:xldn1458pap:10",
    "significance":3,
    "source":"xldn1458pap",
    "source_id":"xldn1458pap",
    "state":9,
    "type":"HTTPDDown"
}
```

### getAlertIds

A GET request that returns the total number of alerts, and a list of the alert IDs for a specified alert filter and a limit.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **query** | String | A JSON or SQL like alert filter. |
| **limit** | Number | The maximum number of alert IDs to return. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object which contains alert details from the following:

| Name | Type | Description |
|---|---|---|
| **total_alerts** | Number | The total number of alerts, or unique alerts. |
| **alert_ids** | Number list | A list of the alert IDs. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getAlertIds" --
data-urlencode 'query=agent!=SYSLOG and description matches "AUTH-
SERVICE"' --data-urlencode 'limit=20'
```

Successful request return:

```
{"total_alerts":20,"alert_ids":[78,234,737,1253,1459,1733,2166,2653,2855,3
133,3414,3538,3729,3905,3991,4110,4160,4536,4692,4701]}
```

### SQL-like Filters

You can now use SQL-like filter conditions similar to URL or Cookbook filters instead of JSON formatted filters:

Example cURL request to get the first 20 alert_ids with query: agent != SYSLOG AND description matches 'AUTH-SERVICE':

```
curl -G -u graze:graze -k -v  "https://localhost/graze/v1/getAlertIds" --
data-urlencode  'query=agent!=SYSLOG and description matches "AUTH-
SERVICE"' --data-urlencode  'limit=20'
```

### removeAlertFromSituation

A POST request that removes a specified alert from a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | The alert ID. |
| **sitn_id** | Number | The Situation ID. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/removeAlertFromSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"alert_id" : 16, "sitn_id" : 7 }'
```

Successful return:

NO RESPONSE TEXT

### resolveAlerts

A POST request that resolves one or more alerts.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **alert_ids** | Number list | Yes | List of IDs of the alerts you want to resolve. |
| **thread_entry_comment** | String | No | Comment you want to add to the resolved alerts. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **status** | Boolean | Whether or not the alerts were resolved. |
| **resolved_alerts** | Number list | List of IDs of alerts that were resolved. |
| **failed_alerts** | Number list | List of IDs of alerts that failed to be resolved. |

Example

Example cURL command to set alerts 45, 76, and 352 as resolved with the comment 'Resolved':

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/resolveAlerts" -H "Content-Type:
application/json; charset=UTF-8" -d '{"alert_ids" : [45,76,352],
"thread_entry_comment" : "Resolved"}'
```

Example return showing that alerts 45, 76 and 352 were successfully resolved and no alerts failed:

```
{"status":true,"resolved_alerts":[45,76,352],"failed_alerts":[]}
```

### setAlertAcknowledgeState

A POST request that acknowledges or unacknowledges the moderator to the alert for a specified alert ID and acknowledge state.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | The alert ID. |
| **acknowledged** | Number | The acknowledge state (0 for unacknowledged, 1 for acknowledged). |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/setAlertAcknowledgeState" -H "Content-Type:
application/json; charset=UTF-8" -d '{"alert_id" : 7, "acknowledged" : 1
}'
```

Successful return:

NO RESPONSE TEXT

### setAlertSeverity

A POST request that sets the severity level for a specified Alert.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **alert_id** | Number | The alert ID. |
| **severity** | Number | The alert's severity as an integer:<br><br>0 Clear<br><br>1 Indeterminate<br><br>2 Warning<br><br>3 Minor<br><br>4 Major<br><br>5 Critical |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. |

| | |
|---|---|
| | For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/setAlertSeverity" -H "Content-Type:
application/json; charset=UTF-8" -d '{"alert_id" : 7, "severity" : 5 }'
```

Successful return:

NO RESPONSE TEXT

## Situations

### addSigCorrelationInfo

A POST request that associates the external client with a specified Situation. This allows Cisco Crosswork Situation Manager to filter events and send only those of interest to an external system.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |
| **service_name** | String | The name of the external service (for example, ServiceNow). |
| **resource_id** | String | The ID of the external service entity to associate with this Situation. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/addSigCorrelationInfo" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 3, "service_name" : "my
service 7", "resource_id" : "my resource 7"}'
```

Successful return:

NO RESPONSE TEXT

### addSituationCustomInfo

A POST request that adds and merges custom information for a specified Situation.

Note:

This method has superseded the now deprecated method **addCustomInfo**.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |

| | | |
|---|---|---|
| **custom_info** | JSON Object | A JSON Object containing the custom information. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/addSituationCustomInfo" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 5, "custom_info" : {
"field1" : "value1" , "field2" : "value2" , "field3" :
["item1","item2","item3"] , "field4" : {"field4-1" : "value4-1","field4-2"
: "value4-2"} }}'
```

Successful return:

NO RESPONSE TEXT

### addThreadEntry

A POST request that adds a new entry to an existing thread in a Situation.

Threads are comments or 'story activity' on Situations.

This endpoint returns the entry ID of the newly created thread entry.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | Situation ID. |
| **thread_name** | String | Name of the existing thread. |
| **entry** | String | Description of the new entry you want to add to the thread. For example, 'And another thing...'. |
| **resolving_step** | Boolean | Whether or not the thread entry you are adding is a resolving step. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **entry_id** | Number | ID of the new thread entry. |

Example

Example cURL request to add a new entry to thread "Support" in Situation 3:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/addThreadEntry" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 3, "thread_name" :
"Support", "entry" : "Test Entry", "resolving_step" : true}'
```

Successful request return providing the ID of the thread entry that has been created:

```
{"entry_id":27}
```

### assignAndAcknowledgeSituation

A POST request that assigns and acknowledges the moderator to the Situation for a specified situation ID and user ID.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request |
| **sitn_id** | Number | The Situation ID |
| **user_id** | Number | The User ID |
| **username** | String | A valid username |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

### assignSituation

A POST request that assigns the moderator to the Situation for a specified Situation ID and user ID.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |
| **user_id** | Number | The User ID. |
| **username** | String | A valid username. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/assignSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 7, "user_id" : 3 }'
```

Successful return:

NO RESPONSE TEXT

### assignTeamsToSituation

A POST request that assigns one or more teams to a Situation. Once successfully run, Cisco Crosswork Situation Manager marks the Situation as overridden and the Teams Manager Moolet can no longer modify its team assignment. See Teams Manager Moolet for more information.Teams Manager Moolet

The endpoint replaces any teams previously assigned to the Situation. You can also use it to unassign all teams from a Situation.

Request Arguments

Include either **team_ids** or **team_names**.

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | The Situation ID. |
| **team_ids** | List | A list of team IDs to assign to the Situation. Specify an empty list to unassign all teams from the Situation. |
| **team_names** | List | A list of team names to assign to the Situation. Specify an empty list to unassign all teams from the Situation. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object with one of the following, depending on the input:

| Name | Type | Description |
|---|---|---|
| **team_ids** | List | A list of team IDs assigned to the Situation. |
| **team_names** | List | A list of team names assigned to the Situation. |

Examples

cURL command 1:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/assignTeamsToSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 1 , "team_ids" : [ 1, 2
]}'
```

Return to cURL command 1:

```
{"team_ids" : [ 1,2 ]}
```

cURL command 2:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/assignTeamsToSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 2 , "team_names" : [
"Team1", "Team2" ]}'
```

Return to cURL command 1:

```
{"team_names" : [ "Team1", "Team2" ]}
```

Unassign example:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/assignTeamsToSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 1 , "team_ids" : []}'
```

Return to unassign example:

```
{"team_ids" : []}
```

### closeSituation

A POST request that closes a specified Situation that is currently open, and optionally closes alerts in the Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |
| **resolution** | Number | Determines what to do with the Situation's alerts:<br>0 Close no alerts.<br>1 Close all alerts in this Situation.<br>2 Close only alerts unique to this Situation. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/closeSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 7, "resolution" : 0 }'
```

Successful return:

NO RESPONSE TEXT

### createSituation

A POST request that creates a manual Situation. The Situation description is set with the **description** parameter. The following Situation settings are pre-set and not configurable here:

- Status: Opened
- Moderator: none assigned

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **description** | String | The new Situation description. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object with the following:

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | The new Situation ID. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/createSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"description" : "This is my
description 12345"}'
```

Successful request return:

```
{"sitn_id":2300}
```

### createThread

A POST request that creates a new thread for a specified Situation.

Threads are comments or 'story activity' on Situations.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The ID of the Situation having a new thread created. |
| **thread_name** | String | The name for the new thread. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/createThread" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 1, "thread_name" : "My
thread 0958"}'
```

Successful return:

NO RESPONSE TEXT

### createThreadEntry

Note

This endpoint has been superseded. Use addThreadEntry instead. All new functionality will be delivered in addThreadEntry.

A POST request that creates a new entry in an existing thread in a Situation.

Threads are comments or 'story activity' on Situations.

This endpoint returns a Boolean indicating whether or not the thread entry was created successfully.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | Situation ID. |
| **thread_name** | String | Name of the existing thread. |
| **entry** | String | Description of the new entry you want to create in the thread. For example, 'And |

| | | |
|---|---|---|
| | | another thing...'. |
| **resolving_step** | Boolean | Whether or not the thread entry you are creating is a resolving step. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Type | Description |
|---|---|
| Boolean | Whether or not the new thread entry was created successfully. |

Example

Example cURL request to create a new entry in thread "Support" in Situation 3:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/createThreadEntry" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 3, "thread_name" :
"Support", "entry" : "Test Entry", "resolving_step" : true}'
```

Successful request return showing that the new thread entry was successfully created:

```
true
```

### deassignSituation

A POST request that that de-assigns the current moderator from the Situation for a specified Situation ID.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/deassignSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 7}'
```

Successful return:

NO RESPONSE TEXT

### getActiveSituationIds

A GET request that returns the total number of active Situations, and a list of their Situation IDs. Active Situations are those that are not Closed, Resolved or Dormant.

Request Argument

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |

There are no other arguments, as this method returns data on all active Situations.

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **total_situations** | Number | The total number of active Situations. |
| **sitn_ids** | Number list | A list of the active Situation IDs. |

Example

Successful request return:

```
{
   "total_situations":10,
   "sitn_ids":[4, 5, 6, 12, 14, 15, 16, 17, 18, 19]
}
```

### getResolvingThreadEntries

A GET request that returns thread entries for a specified Situation that have been marked as resolving steps. Threads are comments or 'story activity' on Situations. Operators can mark one or more thread entries as steps that were used to resolve a Situation.

You can select specific thread entries to return using **start** and **limit** values. If not, their default values return the first 100 entries. The entries returned are ordered by most recent first.

Request Arguments

| Name | Type | Required |
|---|---|---|
| **auth_token** | String | No |
| **sitn_id** | Number | Yes |
| **start** | Number | No |
| **limit** | Number | No |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | Situation ID. |
| **resolving_entries** | List | A list of resolving thread entries. See below. |

The **resolving_entries** list contains the following sub-parameters:

| Name | Type | Description |
|---|---|---|
| **entry_text** | String | Text of the resolving entry. |

| | | |
|---|---|---|
| **user_id** | Number | ID of the user that created the resolving entry. |
| **thread_name** | String | Name of the thread that the resolving entry is in. |
| **time** | Number | Timestamp (in Unix epoch time) of when the resolving entry was created. |
| **entry_id** | Number | ID of the resolving thread entry. |

Example

Example cURL command requesting the first 100 resolving thread entries in Situation 358:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getResolvingThreadEntries" --data-urlencode
"sitn_id=358"
```

Example cURL command requesting the first 10 resolving thread entries in Situation 358:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getResolvingThreadEntries" --data-urlencode
"sitn_id=358" --data-urlencode "start=0" --data-urlencode "limit=10"
```

Successful request return showing the three resolving thread entries in Situation 358:

```
{"sitn_id":358,
"resolving_entries":
[
{"entry_text":"hah","user_id":3,"thread_name":"Support","time":1549387456,
"entry_id":1722},
{"entry_text":"asdfdsf","user_id":3,"thread_name":"Support","time":1549385
762,"entry_id":1721},
{"entry_text":"sdfsdf\n","user_id":3,"thread_name":"Support","time":154938
5747,"entry_id":1720}
]
}
```

### mergeSituations

A POST request that merges multiple specified Situations. You can configure whether to the new Situation supersedes the original Situations or not using the **supersede_original** parameter.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **situations** | Integer | A comma-separated list of the Situations you want to merge. |
| **supersede_original** | Boolean | Determines whether the original merged Situations are superseded by the new Situation. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object with the following:

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | The new Situation ID. |

Example

A cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/mergeSituations?auth_token=c4316d2cac524b96a1e
4c787b68f7e3f&situations=%5B31%2C32%2C33%5D&supersede_original=false"
```

Successful request return:

```
{"sitn_id":30}
```

### getPrcLabels

A GET request that retrieves probable root cause (PRC) information for all alerts or specified alerts within a Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Integer | The Situation ID. |
| **alert_ids** | Number list | A list of the alert IDs (optional). |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getPrcLabels?sitn_id=1&alert_ids=[1,2,3,4]"
```

Successful return:

```
{
   "non_causal":
     [2,3],
   "unlabelled":
     [4],
   "causal":
     [1]
}
```

### getResolvingThreadEntries

A GET request that returns thread entries that have been marked as resolving steps for a specified Situation. Threads are comments or 'story activity' on Situations. Operators can mark one or more thread entries as resolving steps.

You can select specific thread entries to return using **start** and **limit** values. If not, their default values return the first 100 entries. The entries returned are ordered by most recent first.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | Situation ID. |
| **start** | Number | Number of the first thread resolving entry to return (default = 0). Optional. |
| **limit** | Number | Maximum number of resolving thread entries to return (default = 100). Optional. |

Return Parameters

| Type | Description |
|---|---|

| | |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | Situation ID. |
| **resolving_entries** | List | A list of resolving thread entries. See below. |

The **resolving_entries** list contains the following sub-parameters:

| Name | Type | Description |
|---|---|---|
| **entry_text** | String | Text of the resolving entry. |
| **user_id** | Number | ID of the user that created the resolving entry. |
| **thread_name** | String | Name of the thread that the resolving entry is in. |
| **time** | Number | Timestamp (in Unix epoch time) of when the resolving entry was created. |
| **entry_id** | Number | ID of the resolving thread entry. |

Example

Example cURL command requesting the first 100 resolving thread entries in Situation 358:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getResolvingThreadEntries" --data-urlencode
"sitn_id=358"
```

Example cURL command requesting the first 10 resolving thread entries in Situation 358:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getResolvingThreadEntries" --data-urlencode
"sitn_id=358" --data-urlencode "start=0" --data-urlencode "limit=10"
```

Successful request return showing the three resolving thread entries in Situation 358:

```
{"sitn_id":358,
"resolving_entries":
[
{"entry_text":"hah","user_id":3,"thread_name":"Support","time":1549387456,
"entry_id":1722},
{"entry_text":"asdfdsf","user_id":3,"thread_name":"Support","time":1549385
762,"entry_id":1721},
{"entry_text":"sdfsdf\n","user_id":3,"thread_name":"Support","time":154938
5747,"entry_id":1720}
]
}
```

### getSigCorrelationInfo

A GET request that retrieves all correlation information related to a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |

Return Parameter

| Type | Description |
|---|---|

| | |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X GET -u graze:graze -k -v
"https://localhost/graze/v1/getSigCorrelationInfo?sitn_id=5" -H "Content-
Type: application/json; charset=UTF-8"
```

Successful return:

```
 [
     {
     "sig_id": 1,
     "service_name": "Example1",
     "external_id": "Example1",
     "properties": "{"Example1":"Example1"}
     },
     {
     "sig_id": 2,
     "service_name": "Example2",
     "external_id": "Example2",
     "properties": "{"Example2":"Example2"}
     }
   ]
```

### getSimilarSituationIds

A GET request that returns the total number of similar Situations, and a list of their Situation IDs, for a specified Situation filter and a limit.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | ID of the Situation that you want to retrieve similar Situations for. |
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **limit** | Number | Maximum number of Situation IDs to return. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **similiarity_ids** | Number list | List of IDs of the similar Situations. |
| **sig_id** | Number | ID of the Situation that you requested to retrieve similar Situations for. |

Example

Example cURL request to get the first 10 Situation IDs that are similar to Situation ID 1043:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSimilarSituationIds" --data-urlencode
'sitn_id=1043' --data-urlencode 'limit=10'
```

Successful request return:

```
{"similarity_ids":[43,156,177,221,576,1026,1327], "sig_id":1043}
```

### getSimilarSituations

A GET request that returns the details of similar Situations for a specified Situation, a filter and a limit.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | ID of the Situation that you want to retrieve similar Situations for. |
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **limit** | Number | Maximum number of Situation IDs to return. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **similarities** | Array | A list with similarity details. For each similar Situation:<br><br>**sim_sig_id**: ID of the similar situation.<br><br>**similarity**: Similarity value.<br><br>**resolving_steps_count**: Number of resolving steps that the similar situation has. |
| **similar_count** | Number | Number of similar Situations. |
| **sig_id** | Number | ID of the Situation that you requested to retrieve similar Situations for. |

Example

Example cURL request to get the first 20 Situation IDs that are similar to Situation ID 38:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSimilarSituations" --data-urlencode
'"sitn_id"=38' --data-urlencode 'limit=20'
```

Successful request return:

```
{"similarities":[{"sim_sig_id":39,"similarity":1.0,"resolving_steps_count"
:0},{"sim_sig_id":40,"similarity":1.0,"resolving_steps_count":0}],"similar
_count":2,"sig_id":38}
```

### getSituationActions

A GET request that returns the actions for Situations, ordered most recent last. You can use the **from** and **to** arguments to specify a period that you want to retrieve Situation actions for. If you do not specify these, actions for all dates and times are returned.

Request Arguments

| Name | Type | Required |
|---|---|---|
| **auth_token** | String | No |
| **sitn_ids** | Array | Yes |

| | | |
|---|---|---|
| **start** | Integer | Yes |
| **limit** | Integer | Yes |
| **actions** | Array | No |
| **from** | Number | No |
| **to** | Number | No |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing an array of the following:

| Name | Type | Description |
|---|---|---|
| uid | Number | User ID. |
| action_code | Number list | Code for the action in the JSON object. See Situation Action Codes for a list of action codes and their descriptions. |
| description | String | Description of the action. |
| details | String | Details of the action. |
| type | String | Type of action. |
| sig_id | Integer | The Situation ID. |
| timed_at | Integer | Time stamp of the action. |

Examples

Example cURL command to retrieve the first three actions for Situations 1, 2, 3 and 6:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationActions" --data-urlencode
'sitn_ids=[1, 2, 3, 6]' --data-urlencode 'actions=[1]' --data-urlencode
'limit=3' --data-urlencode 'start=0'
```

Example cURL command to retrieve the first 50 actions for Situations 1, 2, 3 and 6 between Unix epoch times 1553861746 and 1553872546:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationActions" --data-urlencode
'sitn_ids=[1, 2, 3, 6]' --data-urlencode 'actions=[1]' --data-urlencode
'limit=50' --data-urlencode 'start=0' --data-urlencode 'from=1553861746' -
-data-urlencode 'to=1553872546'
```

Successful request return:

```
  "activities": [{
      "uid": 2,
      "action_code": 1,
      "description": "Situation Created",
      "details": {},
      "type": "event",
      "sig_id": 1,
      "timed_at": 1507039842
      }, {
      "uid": 2,
      "action_code": 14,
```

```
"description": "Added Alerts To Situation",
"details": {}
"alerts": [1, 2]
}]
}
```

### getSituationAlertIds

A GET request that returns the total number of alerts, and a list of the alert IDs for a specified Situation. This can be either all alerts or just those alerts unique to the Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |
| **for_unique_alerts** | Boolean | Indicates the alerts to get from the Situation:<br>**true** = get only alerts unique to the Situation.<br>**false** = get all alerts in the Situation (default). |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **total_alerts** | Number | The total number of alerts, or unique alerts. |
| **alert_ids** | Number list | A list of the alert IDs. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationAlertIds"     --data-urlencode
"sitn_id=1" --data-urlencode "for_unique_alerts=false"
```

Successful request return:


```
{"total_alerts":232,"alert_ids":[6,10,17,19,22,26,27,29,32,43,44,45,47,52,
67,68,79,81,83,84,96,102,105,108,109,111,113,115,116,125,135,136,138,140,1
42,143,147,151,152,153,165,175,177,178,180,181,188,192,193,207,211,213,217
,223,225,232,238,239,240,244,255,258,259,269,270,272,274,284,293,303,314,3
18,335,357,363,369,374,375,388,398,414,428,430,434,442,443,448,449,450,479
,480,485,486,492,494,504,505,510,511,518,521,529,556,558,563,570,580,594,5
96,599,601,603,628,655,656,661,664,674,684,691,705,714,715,719,720,728,732
,734,750,776,777,781,788,794,808,819,830,835,838,844,857,858,860,861,877,8
82,885,887,890,892,893,900,901,906,912,914,918,926,936,937,959,971,972,984
,994,1004,1013,1016,1019,1020,1023,1033,1043,1045,1068,1076,1082,1083,1085
,1099,1119,1124,1135,1137,1143,1147,1171,1185,1201,1207,1217,1225,1231,123
8,1254,1271,1272,1274,1280,1282,1290,1292,1301,1320,1321,1322,1324,1326,13
27,1331,1332,1333,1362,1379,1402,1414,1423,1433,1443,1454,1468,1472,1473,1
481,1491,1510,1512,1517,1520,1522,1532,1534]}
```


### getSituationDescription

A GET request that returns the description for a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| sitn_id | Number | The Situation ID. |
| description | String | The text in the Situation's description field. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationDescription" --data-urlencode
'sitn_id=1'
```

Successful request return:

```
{"sitn_id" : "1", "description" : "SyslogLamCookbook source"}
```

### getSituationHosts

A GET request that returns the hosts for a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |
| **for_unique_alerts** | Boolean | Optional setting to return hosts for Situations with unique alerts. Defaults to false. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| hosts | JSON Object | An array of all hosts that sent alerts contained in the specified Situation. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationHosts" --data-urlencode
'sitn_id=1'
```

Successful request return:

```
{
   hosts:[
      "xldn1204pap",
      "xldn1215pap",
      "xldn1220pap",
      "vxldn1230pap",
      "xldn1241pap",
      "xldn1252pap",
      "xldn1271pap",
      "xldn1278pap",
      "xldn1297pap",
      "xldn1299pap"
   ]
}
```

### getSituationDetails

A GET request that returns the details for a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| hosts | JSON Object | An array of the details for the specified Situation. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationDetails" --data-urlencode
'sitn_id=1'
```

Successful request return:

```
{
   "category":"Detected",
   "created_at":1415814620,
   "custom_info":null,
   "description":"Sigaliser situation",
   "first_event_time":1415814600,
   "internal_priority":0,
   "last_event_time":1415814619,
   "last_state_change":1415868947,
   "moderator_id":2,
   "sitn_id":3,
   "status":1,
   "story_id":3,
   "superseded_by":null,
   "total_alerts":1403,
   "total_unique_alerts":1403
}
```

### getSituationIds

A GET request that returns the total number of Situations, and a list of their Situation IDs, for a specified Situation filter and a limit.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **query** | String | A JSON or SQL like Situation filter. |
| **limit** | Number | Maximum number of Situation IDs to return. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| total_situations | Number | The total number of Situations, or unique Situations. |
| sitn_ids | Number list | A list of the Situation IDs. |

Example

Example cURL request to get the first 20 Situation IDs with query:description = 'test1' or queue = 5:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getSituationIds"
--data-urlencode 'query=description="test1" or queue = 5' --data-urlencode
'limit=20'
```

Successful request return:

```
{"total_situations":7,"sitn_ids":[87,121,128,278,523,1003,1519]}
```

### getSituationProcesses

A GET request that returns a list of process names for a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| auth_token | String | A valid **auth_token** returned from the **authenticate** request. |
| sitn_id | Number | The Situation ID. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **processes** | List | A list of the Situation's process names. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationProcesses"     --data-urlencode
'sitn_id=1'
```

Successful request return, with a primary process name defined:

```
{"processes":["Knowledge Management","Online Transaction Processing","Web
Content Management","40GbE","8-bit Unicode Transcoding Platform"]}
```

### getSituationServices

A GET request that returns a list of external service names for a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **services** | List | A list of the Situation's service names. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationServices"     --data-urlencode
'sitn_id=1'
```

Successful request return, with a primary service name defined:

```
{"services":["Cloud Management Platform","Geographic Information
Systems","Knowledge Management","Online Transaction Processing","Storage
Subsystem","Web Content Management","0-bit Emulation","40GbE","8-bit
Unicode Transcoding Platform"]}
```

### getSituationTopology

A GET request that returns a JSON object in NetJSON format that represents nodes affected by the Situation.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **sitn_id** | Number | Yes | Situation ID. |
| **context** | Number | No | Number, between 0 and 4, of contextual hops from the nodes directly affected within the Situation to other nodes to be included in the returned object. See 'Vertex Entropy' in *Clustering Algorithm Guide* for more information on contextual hops.Vertex Entropy<br><br>**0** = only nodes directly affected by the Situation. Default.<br><br>**4** = nodes that are up to four hops away from the nodes directly affected by the Situation. |

| | | | |
|---|---|---|---|
| **properties** | List of strings | No | List of the node properties to be returned. Valid properties are:<br>**severity**: Refer Severity in *Implementor Guide* of the node.<br>**prc**: Whether this node is the probable root cause of the alert.<br>**service**: Service affected by the node.<br>**context**: Number of contextual hops between this node and a node directly affected by the Situation. A context of **0** means that the node is directly affected.<br>**description**: Description of the node.<br>**vertex_entropy**: Vertex Entropy of the node. |
| **field_name** | String | No | Attribute of the alert that defines the node. The default is the alert 'source' but you can specify any valid alert field, including **custom_info** attributes. For example, **field_name=custom_info.eventDetails.line**. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br>For codes, see HTTP Status and Error Codes. |

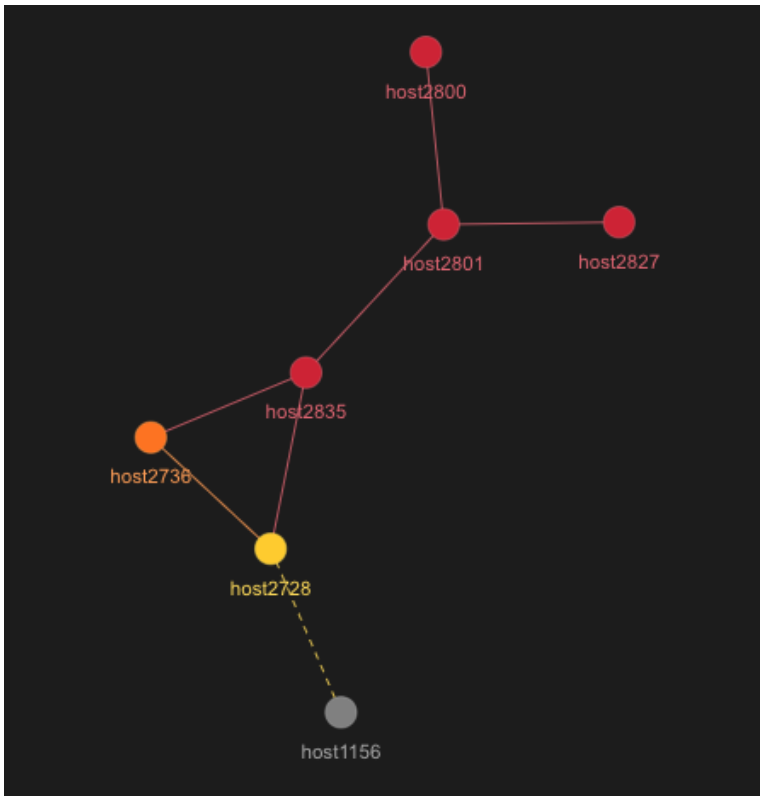
Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **links** | Integer | List of links associated with the Situation. |
| **links.source** | String | Source node of the link. |
| **links.target** | String | Target node of the link. |
| **nodes** | Array | Array of nodes associated with the Situation and their properties. The nodes included depends on the setting of the request property **context**. |
| **nodes.id** | String | Node ID |
| **nodes.properties** | Array | Object containing the properties requested. |

See http://netjson.org/ for more information on the topology data format.

Example

The following topology diagram shows the nodes affected by Situation ID 14, with a context of **1**. In this example, each node represents a host in a network and the Situation represents a network outage. It shows six nodes directly affected by the Situation, their color depending on their severity, and one node which is one hop away, shown in gray.




The following cURL request demonstrates a request to return nodes affected by the Situation and nodes that are one hop away. The returned object is to contain the properties of severity, Vertex Entropy, Probable Root Cause (PRC), service, anddescription.

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationTopology" --data-urlencode
"sitn_id=14" "context=1"
"properties"=["severity","vertex_entropy","prc","service","description"]
```

The successful response returns the following topology information for this Situation. The response always returns the node names in lower case. Note that there is no PRC value for the node that is not directly affected by the Situation. In this example, consider investigating node "host2835" as the cause of the Situation because it has a high severity and a high PRC.


```
{
    "links": [
        {
            "source": "host2728",
            "target": "host2736"
        },
        {
            "source": "host2728",
            "target": "host1156"
        },
        {
            "source": "host2835",
            "target": "host2728"
        },
        {
            "source": "host2801",
            "target": "host2827"
        },
        {
            "source": "host2800",
            "target": "host2801"
        },
```

```
            {
                "source": "host2801",
                "target": "host2835"
            },
            {
                "source": "host2835",
                "target": "host2736"
            }
        ],
        "nodes": [
            {
                "id": "host2835",
                "properties": {
                    "severity": 5,
                    "prc": 0.9862626716344282,
                    "service": "",
                    "context": 0,
                    "description": "",
                    "vertex_entropy": 0.1794592472207979
                }
            },
            {
                "id": "host2736",
                "properties": {
                    "severity": 4,
                    "prc": 0.42722191049803876,
                    "service": "",
                    "context": 0,
                    "description": "",
                    "vertex_entropy": 0.08976540495989357
                }
            },
            {
                "id": "host2728",
                "properties": {
                    "severity": 3,
                    "prc": 0.007672752075071621,
                    "service": "",
                    "context": 0,
                    "description": "",
                    "vertex_entropy": 0.1794592472207979
                }
            },
            {
                "id": "host2827",
                "properties": {
                    "severity": 5,
                    "prc": 0.4262762946261391,
                    "service": "",
                    "context": 0,
                    "description": "",
                    "vertex_entropy": 0.05343516483103129
                }
            },
            {
                "id": "host2801",
                "properties": {
                    "severity": 5,
                    "prc": 0.42722511225514104,
                    "service": "",
                    "context": 0,
                    "description": "",
                    "vertex_entropy": 0.23927899629439717
                }
```

```
        },
        {
            "id": "host2800",
            "properties": {
                "severity": 5,
                "prc": 0.4269879766269776,
                "service": "",
                "context": 0,
                "description": "",
                "vertex_entropy": 0.05343516483103129
            }
        },
        {
            "id": "host1156",
            "properties": {
                "severity": null,
                "prc": null,
                "service": "",
                "context": 1,
                "description": "",
                "vertex_entropy": 0.05343516483103129
            }
        }
    ]
}
```

### getSituationVisualization

A GET request that returns information on the origin and cause of a specified Situation.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **auth_token** | String | Yes | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | Yes | Situation ID. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **sig_id** | Integer | Situation ID. |
| **origin** | String | Process that caused the Situation to be created, for example, **cookbook** or **manual_merge**. |
| **cause** | Object | Cause of the Situation. This varies depending on how the Situation was created. |

Examples

A cURL request to return the information on the origin and cause of Situation ID 358:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationVisualization" --data-urlencode
"sitn_id=358"
```

A successful response for a Situation created by a Cookbook Recipe returns the following information:

```
{
    "origin": "cookbook",
```

```
        "cause": {
            "cookbook_name": "Default Cookbook",
            "recipe_id": 4,
            "cookbook_id": 7,
            "recipe_name": "Recipe 1",
            "reference_event_id": 6
        },
        "sig_id": 1
    }
```

A successful response for a manually created Situation returns the following information:

```
{
    "origin": "Manual Creation",
    "cause": {"uid": 3},
    "sig_id": 6
}
```

A successful response when two Situations have been merged returns the following information:

```
{
    "origin": "Manual Merge",
    "cause": {
        "uid": 3,
        "merged_sigs": [
            8,
            7
        ]
    },
    "sig_id": 9
}
```

If there is no Situation visualization data, the response returns the following information:

```
{
    "additional": {
        "debugMessage":
"com.moogsoft.servletutils.CGeneralServerException:
com.moogsoft.services.CGeneralServiceException: No visualize data found
for Situation ID [2323]"
    },
    "message": "Internal server error",
    "statusCode": 1000
}
```

### getThreadEntries

A GET request that returns thread entries for a specified Situation. Threads are comments or 'story activity' on Situations.

You can select specific thread entries to return using **`start`** and **`limit`** values. If not, their default values return the first 100 entries. The entries returned are ordered by most recent first.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **`auth_token`** | String | A valid **`auth_token`** returned from the **`authenticate`** request. |
| **`sitn_id`** | Number | Situation ID. |
| **`thread_name`** | String | Name of the thread to get entries from. |
| **`start`** | Number | Number of the first thread entry to return (default = 0). |
| **`limit`** | Number | Maximum number of thread entries to return (default = 100). |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **entries** | List | A list of thread entries. See below. |
| **sitn_id** | Number | Situation ID. |
| **thread_name** | String | Name of the thread that the entries are from. |

The **entries** list contains the following sub-parameters:

| Name | Type | Description |
|---|---|---|
| **entry_text** | String | Text of the entry. |
| **user_id** | Number | User ID of the user that created the entry. |
| **time** | Number | Timestamp (in Unix epoch time) of when the entry was created. |
| **entry_id** | Number | ID of the thread entry. |

Example

Example cURL command requesting the first 10 thread entries on thread "Support" in Situation 358:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getThreadEntries"
--data-urlencode "sitn_id=358" --data-urlencode "thread_name=Support" --
data-urlencode "start=0" --data-urlencode "limit=10"
```

Successful request return showing the two thread entries on thread "Support" in Situation 358:

```
{"entries":
[
{"entry_text":"Test Entry","user_id":4,"time":1549455051, "entry_id" : 2},
{"entry_text":"Test Entry",  "user_id":4, "time":1549455053, "entry_id" :
1}
],
"sitn_id":358,"thread_name":"Support"
}
```

### getTopPrcDetails

A GET request that retrieves the top most likely causal alerts, based on their Probable Root Cause value, for a specified Situation ID.

You can select the maximum number of causal alerts to return using a **limit** value. If not specified, the endpoint only returns the alert with the highest root cause probability.

The entries returned are ordered with the highest root cause probability first, irrespective of whether they have been labelled causal or are unlabelled. Alerts marked as symptoms are excluded from the return.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **sitn_id** | Integer | Yes | ID of the Situation you want to retrieve the Probable Root Cause details for. |
| **limit** | Integer | No | Maximum number of causal or unlabelled alerts to return. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing multiple sets of the following data:

| Name | Type | Description |
|---|---|---|
| **rc_probability** | Number | Root cause probability of the alert. |
| **rc_label** | Integer | Label defining whether the alert is causal or not:<br><br>1 = causal<br><br>0 = unlabelled<br><br>-1 = symptom |
| **description** | String | Description of the alert. |
| **alert_id** | Integer | Alert ID. |

Request Example

A cURL request to retrieve the top three causal alerts with the highest root cause probability in Situation 145:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getTopPrcDetails"
--data-urlencode 'sitn_id=145' --data-urlencode 'limit=3'
```

Response Example

A successful response returns the top three causal or unlabelled alerts:

```
{
    "alerts": [
        {
            "rc_probability":0.9933107459030244,
            "description":"Web Server HTTPD is DOWN",
            "rc_label":1,
            "alert_id":53
        },
        {
            "rc_probability":0.9933092393241993,
            "description":"Web Server HTTPD is DOWN",
            "rc_label":1,
            "alert_id":8
        },
        {
            "rc_probability":0.22480057080448923,
            "description":"Web Server HTTPD is DOWN",
            "rc_label":0,
            "alert_id":39
        }
    ]
}
```

### removeSigCorrelationInfo

A DELETE request that removes all correlation information related to a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |

| | | |
|---|---|---|
| **serviceName** | String | The service name (optional). |
| **externalId** | String | The external ID (optional). |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/removeSigCorrelationInfo" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 3, "service_name" : "my
service 7", "external_id" : "my resource 7"}'
```

A successful return is an HTTP 200 response.

### resolveSituation

A POST request that resolves a specified Situation that is currently open.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/resolveSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 5}'
```

Successful return:

NO RESPONSE TEXT

### setPrcLabels

A POST request that sets the probable root cause (PRC) labels for specified alerts within a Situation. You must specify at least one alert ID and a PRC level for the alert.

You can mark alerts as causal, non_causal or unlabelled within a Situation. An alert can have different PRC levels within different Situations.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | The Situation ID. |
| **alert_ids** | Number list | A list of the alert IDs. |

| | | |
|---|---|---|
| **causal**<br>**non_causal**<br>**unlabelled** | List | PRC levels. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -POST -u graze:graze -k -v "https://localhost/graze/v1/setPrcLabels"
--data-urlencode "sitn_id=1" --data-urlencode "causal=[1]" --data-
urlencode "non_causal=[2,3]" --data-urlencode "unlabelled=[4]"
```

Successful return:

NO RESPONSE TEXT

### setResolvingThreadEntry

A POST request that sets or clears a thread entry in a Situation as a resolving step.

Threads are comments or 'story activity' on Situations.

This endpoint returns a Boolean indicating whether the thread entry was successfully set or cleared as a resolving step.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **entry_id** | String | ID of the thread entry. |
| **resolving_step** | Boolean | Whether you are setting or clearing the thread entry as a resolving step. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Type | Description |
|---|---|
| Boolean | Whether or not the thread entry was successfully set or cleared as a resolving step. |

Example

Example cURL request to set thread entry 28 as a resolving step:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/setResolvingThreadEntry" -H "Content-Type:
application/json; charset=UTF-8" -d '{"entry_id" : 28, "resolving_step" :
true}'
```

Successful request return showing that the thread entry was successfully set as a resolving step:

```
true
```

### setSituationAcknowledgeState

A POST request that acknowledges or unacknowledges the moderator to the Situation for a specified Situation ID and acknowledged state.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | The Situation ID. |
| **acknowledged** | Number | The acknowledge state (0 for unacknowledged, 1 for acknowledged). |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/setSituationAcknowledgeState" -H "Content-
Type: application/json; charset=UTF-8" -d '{"sitn_id" : 64, "acknowledged"
: 1 }'
```

Successful return:

NO RESPONSE TEXT

### setSituationDescription

A POST request that sets the description for a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |
| **description** | String | The description text to be applied to the Situation. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/setSituationDescription" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 6, "description" : "This
is my description 12345"}'
```

Successful return:

NO RESPONSE TEXT

### setSituationProcesses

A POST request that applies a list of processes to a specified Situation.

Any other processes already associated with the Situation are removed.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |
| **process_list** | List | A list of process names as text strings (for example, [**"P1"**,**"P2"**]). If any processes supplied do not exist in the database, they are created and assigned to the Situation. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/setSituationProcesses" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 7, "process_list" :
["Knowledge Management", "90nm Manufacturing"]}'
```

Successful return:

NO RESPONSE TEXT

### setSituationServices

A POST request that applies a list of external services to a specified Situation.

Any other services already associated with the Situation are removed.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID. |
| **service_list** | List | A list of service names as text strings (for example, [**"S1"**,**"S2"**]). If any services supplied do not exist in the database, they are created and assigned to the Situation. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/setSituationServices" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_id" : 8, "service_list" :
["Knowledge Management", "90nm Manufacturing"]}'
```

Successful return:

NO RESPONSE TEXT

## User Management

### applyNewLicense

A POST request that allows a Cisco Crosswork Situation Manager license to be added via Graze.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **license** | String | A valid license key. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/applyNewLicence" -H "Content-Type:
application/json; charset=UTF-8" -d '{"license" : "<your license key>"}'
```

Successful request return:

```
HTTP/2 200
```

authenticate

A GET request that provides the auth_token required by all other Graze API requests which do not provide the basic authentication header. Graze users can have multiple concurrent Graze sessions with the same username and password.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **username** | String | A valid Cisco Crosswork Situation Manager username. |
| **password** | String | The username's corresponding password. |

Return Parameter

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A session ID for use in subsequent requests. |

Example

cURL command:

```
curl -k -v
"https://localhost/graze/v1/authenticate?username=graze&password=graze"
```

Successful request return:

```
{"auth_token":"878b3ec57d464aee80d09893221be8e8"}
```

All requests (other than **authenticate**) require a valid **auth_token** or basic authentication header. Therefore before any Graze API request is used, a valid **authenticate** request must be successfully made unless basic authentication headers are used.

Inactive sessions will be logged out after one hour, and a new **authenticate** request must be made to get a new valid **auth_token**.

Note:

If you are making regular Graze requests within a one hour timeframe you are considered active and your session does not expire.

createTeam

A POST request that creates a new team.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **name** | String | Mandatory - the new team (unique) name. |
| alert_filter | String | The team alerts filter. Either a SQL like filter or an JSON representation of the filter. |
| services | JSON List | List of the team services names or IDs. |
| sig_filter | String | The situation filters. Either a SQL like filter or an JSON representation of the filter. |
| landing_page | String | The team default landing page. |
| active | Boolean | False if the team is inactive, true if the team is active. Default is **true**. |
| description | String | The team description. |
| users | List of numbers or strings | The team users (either IDs or usernames). |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object with the following:

| Name | Type | Description |
|---|---|---|
| **team_id** | Number | The new team ID |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/createTeam"
-H "Content-Type: application/json; charset=UTF-8" -d '{"name" : "my team
name 1", "alert_filter" : "manager = \"my_manager\" and (class =
\"my_class_12345\" or external_id = \"my_ext_12345\")", "services" :
["Identity Management","Yellow Pages"], "sig_filter" : "description =
\"my_description_12345\" or queue = 50", "landing_page" :
{"type":"situations","id":"open"}, "active" : true, "description" : "The
team description 12345", "users" : ["user1","user2","user3"]}'
```

Successful request return:

```
{"team_id":16}
```

**createUser**

A POST request that creates a new user.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **username** | String | Mandatory - the new user (unique) login username. |
| **password** | String | The new user password (only valid for DB realm). |
| **active** | Boolean | true if the user active, false if the user inactive, default to true. |
| **email** | String | The user email address. |
| **fullname** | String | The user full name. |
| **roles** | JSON list | That list should contain either the list the role IDs or the role names, for example, "roles":["Super User"]. |
| **primary_group** | String or Number | The user primary group name or primary group ID. |
| **department** | String or number | The user department ID or name. |
| **joined** | Number | The time the user joined (in Unix time). |
| **timezone** | String | The user timezone. |
| **contact_num** | String | The user phone number. |
| **session_expiry** | Number | The number of minutes after which the user session will expire. Defaults to the system default. |
| **competencies** | JSON list | A list with the user competencies. Each competency should have have name or cid and ranking. That is, something like:<br>`[ {"name":"SunOS", "ranking": 40}, {"name":"SAP", "ranking": 50}, {"name":"EMC", "ranking": 60} ]` |
| **teams** | JSON list of numbers or strings | List of the user teams. The list should contains either the list of the teams ID or the teams name. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object with the following:

| Name | Type | Description |
|---|---|---|
| **user_id** | Number | The new user ID |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/createUser"
-H "Content-Type: application/json; charset=UTF-8" -d '{"username" :
"johndoe1", "roles" : ["Super User", "Operator"], "password" : "johndoe1",
"active" : true, "email" : "johndoe@moogsoft.com", "fullname" : "John
Doe", "primary_group" : "Network", "department" : "Support", "joined" :
1494951621, "timezone" : "Europe/London", "contact_num" : "555-1234",
"session_expiry" : null, "competencies" : [{"name":"SunOS", "ranking":
40},{"name":"SAP", "ranking": 50},{"name":"EMC", "ranking": 60}], "teams"
: ["my team 1","my team 2","my team 3"], "properties" : null}'
```

Successful request return:

```
{"user_id":777}
```

### getTeam

A GET request that returns a team's details by team ID or name.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **team_id** | Integer | The unique ID of the team to retrieve information about. |
| **name** | String | The name of a valid team to retrieve information about. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL commands:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeam?team_id=1"

curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeam?name=Cloud DevOps"
```

Successful request return:

```
{
    "room_id": 1,
    "alert_filter": "",
    "user_ids": [
        3
    ],
    "sig_filter": "",
    "landing_page": null,
    "description": "Example Team",
    "active": true,
    "team_id": 1,
    "services": [],
    "users": [
        "admin"
    ],
    "name": "Cloud DevOps",
    "service_ids": []
}
```

### getTeamsForService

A GET request to return all teams related to the service with the specified ID or name.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid auth_token returned from the authenticate request. |
| **service_id** | String | The ID of the service. |
| name | String | The name of the service. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL commands:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamsForService?service_id=1"

curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamsForService?service_name=web"
```

Successful request return:

```
[
    {
        "room_id": 1,
        "alert_filter": "",
        "user_ids": [
            3
        ],
        "sig_filter": "",
        "name": "Cloud DevOps",
        "landing_page": "",
        "description": "Example Team",
        "active": true,
        "service_ids": [
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
            9,
            10,
            11
        ],
        "team_id": 1,
        "services": [
            "Commerce",
            "Compute",
            "CRM",
            "Database",
            "Mobile",
            "Networking",
```

```
                "Remote",
                "Social",
                "Storage",
                "Switch",
                "Web"
            ],
            "users": [
                "admin"
            ]
        }
    ]
```

### getTeamSituationIds

A GET request that returns the total number of Situations that are assigned to a team, and a list of their Situation IDs.

Request Argument

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **team_name** | String | The name of an existing team. |

There are no other arguments, as this method returns data on all Situations assigned to a team.

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **total_situations** | Number | The total number of Situations assigned to a team. |
| **sitn_ids** | Number list | A list of Situation IDs of the Situations assigned to a team. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamSituationIds" --data-urlencode
"team_name=Cloud Devops"
```

Successful request return:

```
{"total_situations":35,"sitn_ids":[20,21,39,55,85,119,145,180,208,233,244,
275,305,358,460,461,485,518,574,575,592,616,666,695,696,740,800,892,919,96
0,992,993,1027,1047,1092]}
```

### getTeamSituationStats

A GET request that returns the number of active Situations assign to a team over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **teams** | Array | An array of team IDs (optional). If no teams are provided, the endpoint returns data for the top 10 teams. |
| **from** | Number | The time from when the data points will be collected. This is a timestamp from epoch in seconds. |
| **to** | Number | The time until when the data points will be collected. This is a timestamp from epoch in seconds. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the team |
| **datapoints** | Number array | A array containing multiple nested arrays of datapoint (timestamp + value) |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamSituationStats" --data-urlencode
'teams=[1,2]'  --data-urlencode 'from=1513508950' --data-urlencode
'to=1513595370'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
        "target": "Cloud DevOps"
}, {
        "datapoints": [],
        "target": "Database"
}]
```

### getUserInfo

A GET request that returns information about a specified user.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **user_id** | Number | The user ID of the user to get information about. |
| **username** | String | A valid username. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **user_id** | Number | The user's ID. |
| **full_name** | String | The full name of the user. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getUserInfo" --
data-urlencode "user_id=57"
```

Successful request return:

```
{"full_name":"Lonnie Holmes","user_id":57}
```

### getUserRoles

Fetches the user's roles from the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **user_id** | Number | The user's ID. |
| **username** | String | A valid username. |

Return parameter

| Type | Description |
|---|---|
| NativeObject | A Javascript object containing Role ID, Role name and Role description. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getUserRoles" --
data-urlencode "username=bigfish917"
```

Successful request return:

```
[{"id" : 2, "name" : "Administrator", "description" : "Administrator"},
{"id" : 4, "name" : "Operator", "description" : "Operator"}, {"id" : 5,
"name" : "Customer", "description" : "Customer"}]
```

### getUsers

Fetches a list of all users in the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| **auth_token** | String<br>(Mandatory) | A valid **auth_token** returned from the **authenticate** request. |
| limit | Integer<br>(Optional) | The maximum number of results to return. Default is 1000. |

Return parameter

| Type | Description |
|---|---|
| NativeObject | A JSON list of all users, displaying uid, teams, fullname and username. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getUsers" --data-urlencode "limit=3"
```

Successful request return:

```
[
    {
        "uid": 3,
        "teams": [
            "Cloud DevOps"
        ],
        "fullname": "Administrator",
        "username": "admin"
    },
    {
        "uid": 6,
        "teams": [],
        "fullname": "Nagios",
        "username": "Nagios"
    },
    {
        "uid": 5,
        "teams": [],
        "fullname": "Webhook",
        "username": "Webhook"
    }
  ]
```

### getUserTeams

Fetches the team names and IDs associated with the specified user ID or username.

Request Argument

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **user_id** | Number | A valid user ID. |
| **username** | String | A valid username. |

Return parameter

| Type | Description |
|---|---|
| NativeObject | A Javascript object containing Team ID and Team name. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getUserTeams" --data-urlencode "username=admin"
```

Successful request return:

```
[{"id" : 11, "name" : "Team1"}, {"id" : 12, "name" : "Team2"}, {"id" : 2, "name" : "Team3"}, {"id" : 6, "name" : "Team4"}, {"id" : 10, "name" : "Team5"}]
```

### updateTeam

A POST request that modifies an existing team.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| team_id | Number | The team ID (required). |
| name | String | The new team name. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is. |
| alert_filter | String | The new team alerts filter. Either a SQL like filter or an JSON representation of the filter. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is. |
| services | JSON List | List of the team services names or IDs. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is. |
| sig_filter | String | The situation filters. Either a SQL like filter or an JSON representation of the filter. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is. |
| landing_page | String | The team default landing page. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is. |
| active | Boolean | False if the team is inactive, true if the team is active. Default to true. Exclude this atttribute to leave Cisco Crosswork Situation Manager as is. |
| description | String | The team description. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is. |
| users | List of numbers or strings | The team users (either IDs or usernames). Exclude this attribute to leave Cisco Crosswork Situation Manager as it is. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/updateTeam"
-H "Content-Type: application/json; charset=UTF-8" -d '{"team_id" : 16,
"name" : "my team name RENAMED", "active" : true, "description" : "The
team description", "users" : []}'
```

Successful return:

NO RESPONSE TEXT

### updateUser

A POST request that modifies an existing user.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **username** | String | Username for user to be edited. Required (optional if user ID is used). |
| **uid** | Long | User ID for user to be edited. Required (optional if username used). |
| **password** | String | The new user password (only valid for DB realm). Optional. |
| **active** | Boolean | **true** if the user active, **false** if the user inactive. Defaults to **true**. Optional. |

| | | |
|---|---|---|
| **email** | String | User's email address. Optional. |
| **fullname** | String | User's full name. Optional. |
| **roles** | JSON list | A list that should contain either the list of the role IDs or the role names, for example, "roles":["Super User"]. Optional. |
| **primary_group** | String or Number | User's primary group name or primary group ID. Optional. |
| **department** | String or number | User's department ID or name. Optional. |
| **timezone** | String | User's timezone. Optional. |
| **contact_num** | String | User's phone number. Optional. |
| **session_expiry** | Number | Number of minutes after which the user session will expire. Default to system default. Optional. |
| **competencies** | JSON list | A list with the user competencies. Optional. Each competency should have have name or cid and ranking. That is, something like:<br>**[**<br>**{"name":"SunOS", "ranking": 40},**<br>**{"name":"SAP", "ranking": 50},**<br>**{"name":"EMC", "ranking": 60}**<br>**]** |
| **teams** | JSON list of numbers or strings | List of the user teams. The list should contains either the list of the teams ID or the teams name. Optional. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/updateUser"
-H "Content-Type: application/json; charset=UTF-8" -d '{"uid" : 5,
"active" : true, "password" : "test", "roles" : ["Super User",
"Operator"], "teams" : ["my team 1"], "competencies" : [{"name":"SunOS",
"ranking": 40},{"name":"SAP", "ranking": 50},{"name":"EMC", "ranking":
60}], "session_expiry" : null, "properties" : null, "contact_num" : "555-
123456", "timezone" : "Europe/London", "fullname" : "John Doe",
"department" : "Support", "primary_group" : "Network", "email" :
"test@test.com"}'
```

Successful request return:

NO RESPONSE TEXT

## Security Realms

### createSecurityRealm

A POST request that creates a new security realm from an Identity Provider (IdP) URL. The request also adds the realm configuration you provide.

Warning:

Warn any users who logged into Cisco Crosswork Situation Manager using the default realm before using this request. The system may log out users when the new realm becomes active.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **name** | String<br><br>(Required) | Name of the security realm. |
| **type** | String<br><br>(Required) | Security realm type. This must be **"SAML2"**. |
| **active** | Boolean<br><br>(Required) | Determines whether the new realm is active in Cisco Crosswork Situation Manager on creation.<br><br>You can create an inactive realm for testing purposes. For example, you can verify if a security realm with that name already exists or if it fails. |
| **configuration** | JSON Object<br><br>(Required) | JSON object containing the realm configuration. For information on the configuration properties, see Security Configuration Reference.Security Configuration Reference<br><br>Upload your IdP metadata file using idpMetadata or specify the location of the file using idpMetadataUrl. For example:<br><br>"idpMetadataUrl":"http://<location_of_idp_metadata>"<br><br>"idpMetadata":"<raw_ipd_metadata.xml>" |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command example to create a SAML realm that maps users to the default user mappings and has a maximum authentication lifetime of 60 seconds:

```
curl -X POST \
        -u graze:graze \
        -k -v "https://localhost/graze/v1/createSecurityRealm" \
        -d {"name"="mySamlRealm",
"type"="SAML2","active="true","configuration"=
        {

"idpMetadataUrl":"http://exampleIdP:18080/auth/realms/master/protocol/saml
/descriptor",
        "defaultRoles":["Operator"],
        "defaultTeams":["Cloud DevOps"],
        "defaultGroup":"End-User",
        "existingUserMappingField":"username",
        "username":"$username",
        "email":"$email",
        "fullname":"$firstname $lastname",
        "maximumAuthenticationLifetime":60
        }
}
```

A successful request returns an HTTP200response.

### getSecurityRealm

A GET request that returns a JSON object containing the names and configuration details of active security realms.

Request Arguments

None required.

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command to return the configuration of any active security realm in Cisco Crosswork Situation Manager:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSecurityRealms"
```

Successful requests return a JSON object representing the active realms. The following example shows a test SAML realm and a Google realm:

```
{
        "Test Saml Realm": {
                "configuration":  {
                        "defaultGroup":"EndUser",
                        "realmType":"SAML2",
                        "existingUserMappingField":"username",

"spMetadataFile":"/usr/share/moogsoft/config/keycloak.my_sp_metadata.xml",
                        "defaultRoles":["Operator"],
                        "defaultTeams":["Cloud DevOps"],
                        "fullname":"$FirstName $LastName",
                        "email":"$Email",

"idpMetadataFile":"/usr/share/moogsoft/config/keycloak.my_idp_metadata.xml
",
                        "username":"$Email",
                        "maximumAuthenticationLifetime":60},
                        "name":"Test Saml Realm",
                        "active":true,
                        "type":"SAML2"
                                        }
        ,"Google realm": {
                "configuration": {
                        "realmType":"GOOGLE"},
                        "name":"Google realm",
                        "active":true,"type":"GOOGLE"}
}
```

### updateSecurityRealm

A POST request that updates an existing security realm in the database.

Warning

Warn any users who logged into Cisco Crosswork Situation Manager using the default realm before using this request. The system may log out users when the updated realm becomes active.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **name** | String<br><br>(Required) | Name of the security realm. |
| **type** | String | Security realm type. This must be **"SAML2"**. |
| **active** | Boolean | Determines whether the new realm is active or not. |
| **configuration** | JSON Object | JSON object containing the realm configuration. You must include all mandatory configuration properties; otherwise the request returns an error. For information on the configuration properties, see Security Configuration Reference.Security Configuration Reference |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command to update a SAML realm with a new X509 certificate:

```
curl -X POST \
     -u graze:graze \
     -k -v "https://localhost/graze/v1/updateSecurityRealm" \
     -d {"name"="mySamlRealm"
     -d 'configuration=
     {
         "idpMetadata":"<?xml version=\"1.0\" encoding=\"UTF-
8\"?>\r\n<EntitiesDescriptor Name=\"urn:keycloak\"
xmlns=\"urn:oasis:names:tc:SAML:2.0:metadata\"\r\nxmlns:dsig=\"http:\/\/ww
w.w3.org\/2000\/09\/xmldsig#\">\r\n<EntityDescriptor
entityID=\"http:\/\/moogsaml:18080\/auth\/realms\/master\">\r\n<IDPSSODesc
riptor
WantAuthnRequestsSigned=\"true\"\r\nprotocolSupportEnumeration=\"urn:oasis
:names:tc:SAML:2.0:protocol\">\r\n<KeyDescriptor
use=\"signing\">\r\n<dsig:KeyInfo>\r\n<dsig:KeyName>l8ddhI8SroeNnlq0TkTxIj
2VI-
0bvr2QfG_o32jWeKI<\/dsig:KeyName>\r\n<dsig:X509Data>\r\n<dsig:X509Certific
ate>MIICmzCCAYMCBgFk8A9vMjANBgkqhkiG9w0BAQsFADARMQ8wDQYDVQQDDAZtYXN0ZXIwHh
cNMTgwNzMxMTExNjQwWhcNMjgwNzMxMTExODIwWjARMQ8wDQYDVQQDDAZtYXN0ZXIwggEiMA0G
CSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCOliZ3dBu696slYduAb1BMuvR1bMdTKVBMICWaEE
cS8Rzw8gWthPQpw2e202LjOeu4VkTVmEEAUa2IrLS4QpYgyhOuzapcIGF4kB0ARebalWa7C9od
9%2BeTqWgvXPrDOkzp7g%2B%2Ba5yvtKxE3ieUORPpACvLWcbkMwyb%2Be5V8%2Bz8n4263Uol
8srSaxLsm\/oTozJNwbG%2BbzV8JQHU3xFV5nFbyNySvc%2B\/B7tDFZuJC5BMu6bwi\/rPqp5
OMcuB1W%2BxCcX7IYPphnBjRWNyQJD3gRCkjrujISkTEcqpZEjR79isbofQaPDi5TSjglPD5rr
0OWMVqv91a1\/pVN2y0y%2BRlT8HAgMBAAEwDQYJKoZIhvcNAQELBQADggEBAAgRhWYKESVsTR
AUVYzHYptd3\/eX47%2BTVXhjPO0ORLUJbHtfhgohtyejd6ohazkcSgMy6%2BwaeVojqq4Q\/t
zCOW2EAqO9QOQdaBWOPxDXhJ9TGQJE2E28SS2Gg6paAMfRmtA7c6xXii%2BYfLo3PG1SSc\/sG
e4KIPKflkqqDEqEeaY1olPZU2bLnpMSIui2nK1crE2%2Bt9apLWAGosah6scMGZ9vTrtOVrNuh
B2LuU3cvRQWrUBaQuXQsBV7Q6a8lkrrZ6rjAIbO4vcEL4yjQpnA%2BhetuhBlGPQj6ntuhdnmo
KmWYY97wk8eXwblhQxg8GUyfqabfOAKwiGAklxgkexm20M=<\/dsig:X509Certificate>\r\
n<\/dsig:X509Data>\r\n<\/dsig:KeyInfo>\r\n<\/KeyDescriptor>\r\n\r\n<Single
LogoutService\r\nBinding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST\"\r\nLocation=\"http:\/\/moogsaml:18080\/auth\/realms\/master\/protoc
ol\/saml\"
\/>\r\n<SingleLogoutService\r\nBinding=\"urn:oasis:names:tc:SAML:2.0:bindi
ngs:HTTP-
Redirect\"\r\nLocation=\"http:\/\/moogsaml:18080\/auth\/realms\/master\/pr
otocol\/saml\" \/>\r\n<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent<\/NameIDFormat>\r\n<NameIDFormat>urn:oasis:names:tc:SAML
:2.0:nameid-
format:transient<\/NameIDFormat>\r\n<NameIDFormat>urn:oasis:names:tc:SAML:
```

```
1.1:nameid-
format:unspecified<\/NameIDFormat>\r\n<NameIDFormat>urn:oasis:names:tc:SAM
L:1.1:nameid-format:emailAddress<\/NameIDFormat>\r\n<SingleSignOnService
Binding=\"urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST\"\r\nLocation=\"http:\/\/moogsaml:18080\/auth\/realms\/master\/protoc
ol\/saml\"
\/>\r\n<SingleSignOnService\r\nBinding=\"urn:oasis:names:tc:SAML:2.0:bindi
ngs:HTTP-
Redirect\"\r\nLocation=\"http:\/\/moogsaml:18080\/auth\/realms\/master\/pr
otocol\/saml\"
\/>\r\n<SingleSignOnService\r\nBinding=\"urn:oasis:names:tc:SAML:2.0:bindi
ngs:SOAP\"\r\nLocation=\"http:\/\/moogsaml:18080\/auth\/realms\/master\/pr
otocol\/saml\"
\/>\r\n<\/IDPSSODescriptor>\r\n<\/EntityDescriptor>\r\n<\/EntitiesDescript
or>",
     "defaultRoles":["Operator"],
     "defaultTeams":["Cloud DevOps"],
     "defaultGroup":"End-User",
     "existingUserMappingField":"username",
     "username":"$username",
     "email":"$email",
     "fullname":"$firstname $lastname",
     "maximumAuthenticationLifetime":60
     }'
```

cURL command to deactivate an active SAML realm:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/updateSecurityRealm" -d "name=mySamlRealm" -d
"active=false"
```

Successful return:

A successful request returns an HTTP 200 response.

## Dashboards and Reporting

### getMTTAStats

A GET request that returns the Mean Time To Acknowledge a situation in the system over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | The time from when the data points will be collected. A timestamp from epoch in seconds. |
| **to** | Number | The time until when the data points will be collected. A timestamp from epoch in seconds. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | **"mtt_acknowledge"** |
| **datapoints** | Number array | An array containing multiple nested arrays of datapoints (timestamp + value). |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getMTTAStats" --
data-urlencode 'from=1513508950' --data-urlencode 'to=1513595370'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
        "target": "mtt_acknowledge"
}]
```

### getMTTDStats

A GET request that returns the Mean Time To Detect a situation in the system over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | The time from when the data points will be collected. A timestamp from epoch in seconds. |
| **to** | Number | The time until when the data points will be collected. A timestamp from epoch in seconds. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | **"mtt_detect"** |
| **datapoints** | Number array | An array containing multiple nested arrays of datapoints (timestamp + value). |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getMTTDStats" --
data-urlencode 'from=1513508950' --data-urlencode 'to=1513595370'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
```

```
        "target": "mtt_detect"
}]
```

### getMTTRStats

A GET request that returns the Mean Time To Resolve a Situation in the system over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | The time from when the data points will be collected. A timestamp from epoch in seconds. |
| **to** | Number | The time until when the data points will be collected. A timestamp from epoch in seconds. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | **"mtt_resolve"** |
| **datapoints** | Number array | A array containing multiple nested arrays of datapoint (timestamp + value) |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getMTTRStats" --
data-urlencode 'from=1513508950' --data-urlencode 'to=1513595370'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
        "target": "mtt_resolve"
}]
```

### getReassignedSituationStats

A GET request that returns the number of situations reassigned in the system over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | Time from when the data points will be collected. A timestamp from epoch in seconds. |
| **to** | Number | Time until when the data points will be collected. A timestamp from epoch in seconds. |

Return Parameters

| Type | Description |
|---|---|

| | |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Reassigned Situation" |
| **datapoints** | Number array | A array containing multiple nested arrays of datapoint (timestamp + value) |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getReassignedSituationStats" --data-urlencode
'from=1513508950' --data-urlencode 'to=1513595370'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
        "target": "Reassigned Situation"
}]
```

### getReoccurringSituationStats

A GET request that returns the percentage of reoccurring situations in the system over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | Time from when the data points will be collected. A timestamp from epoch in seconds. |
| **to** | Number | Time until when the data points will be collected. A timestamp from epoch in seconds. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | **"Reoccurring Situation"** |
| **datapoints** | Number array | An array containing multiple nested arrays of datapoints (timestamp + value). |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getReoccurringSituationStats" --data-urlencode
'from=1513508950' --data-urlencode 'to=1513595370'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
        "target": "Reoccurring Situation"
}]
```

### getServiceSituationStats

A GET request that returns the number of active Situations impacting a service over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **services** | Array | An array of services IDs (optional). If no services are provided, the endpoint returns empty data. |
| **from** | Number | Time from when the data points will be collected.A timestamp from epoch in seconds. |
| **to** | Number | Time until when the data points will be collected.A timestamp from epoch in seconds. |
| **aggregation** | String | Can be one of:<br>**"none"** - No aggregation of results.<br>**"sum"** - Aggregation of all services provided. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the service |
| **datapoints** | Number array | A array containing multiple nested arrays of datapoint (timestamp + value) |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getServiceSituationStats" --data-urlencode
'services=[1,2]' --data-urlencode 'from=1513508950' --data-urlencode
'to=1513595370' --data-urlencode 'aggregation=sum'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
        "target": "Service A"
}, {
        "datapoints": [],
        "target": "Service B"
}]
```

### getSeveritySituationStats

A GET request that returns the number of situations by severity.

Request Argument

| Name | Type | Description |
|---|---|---|
| **severity** | Array | An array of severity IDs. Optional. If not given, returns a default set of severities: Clear, Indeterminate, Warning, Minor, Major, Critical. |
| **from** | Number | Time from when the data points will be collected. A timestamp from epoch in seconds. |
| **to** | Number | Time until when the data points will be collected. A timestamp from epoch in seconds. |
| **aggregation** | String | Can be one of:<br>**"none"** - No aggregation of results.<br>**"sum"** - Aggregation of all severities provided. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the status. |
| **datapoints** | Number array | An array containing multiple nested arrays of datapoints (timestamp + value). |

Example

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSeveritySituationStats" --data-urlencode
'from=1516216020' --data-urlencode 'to=1516282420' --data-urlencode
'severity=[0, 1]' --data-urlencode 'aggregation=sum'
```

Successful request return:

```
[{
        "datapoints": [
                [92.0, 1516008478000],
                [666.0, 1516030078000]
        ],
```

```
        "target": "Major"
}, {
        "datapoints": [
                [1.0, 1515947278000],
                [35.0, 1515958078000],
                [63.0, 1515976078000],
                [241.0, 1515994078000],
                [4.0, 1516015678000]
        ],
        "target": "Minor"
}]
```

### getStatusSituationStats

A GET request that returns the number of situations by status.

Request Argument

| Name | Type | Description |
|---|---|---|
| **status** | Array | An array of status IDs. Optional. If not given, returns a default set of statuses: Opened, Unassigned, Assigned, Acknowledged, Unacknowledged, Resolved. |
| **from** | Number | Time from when the data points will be collected.A timestamp from epoch in seconds. |
| **to** | Number | Time until when the data points will be collected.A timestamp from epoch in seconds. |
| **aggregation** | String | Can be one of:<br>**"none"** - No aggregation of results.<br>**"sum"** - Aggregation of all statuses provided. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the status. |
| **datapoints** | Number array | An array containing multiple nested arrays of datapoints (timestamp + value). |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getStatusSituationStats" --data-urlencode
'from=1515943678' --data-urlencode 'to=1516030078' --data-urlencode
'status=[1, 2]' --data-urlencode 'aggregation=sum'
```

Successful request return:

```
[{
        "datapoints": [
                [92.0, 1516008478000],
                [666.0, 1516030078000]
        ],
        "target": "Opened"
}, {
        "datapoints": [
```

```
                [1.0, 1515947278000],
                [35.0, 1515958078000],
                [63.0, 1515976078000],
                [241.0, 1515994078000],
                [4.0, 1516015678000]
        ],
        "target": "Assigned"
}]
```

### getSystemSituationStats

A GET request that returns the number of active Situations in the system over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | The time from when the data points will be collected. A timestamp from epoch in seconds. |
| **to** | Number | The time until when the data points will be collected. A timestamp from epoch in seconds. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | **"System"** |
| **datapoints** | Number array | An array containing multiple nested arrays of datapoints (timestamp + value). |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSystemSituationStats" --data-urlencode
'from=1513508950' --data-urlencode 'to=1513595370'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
        "target": "Open Situations"
}]
```

### getTeamSituationStats

A GET request that returns the number of active Situations assigned to a team over time.

Request Argument

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **teams** | Array | An array of team IDs (optional). If no teams are provided, the endpoint returns empty data. |
| **from** | Number | The time from when the data points will be collected. A timestamp from epoch in seconds. |
| **to** | Number | The time until when the data points will be collected. A timestamp from epoch in seconds. |
| **aggregation** | String | Can be one of:<br>**"none"** - No aggregation of results.<br>**"sum"** - Aggregation of all teams provided. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the team. |
| **datapoints** | Number array | A array containing multiple nested arrays of datapoint (timestamp + value). |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamSituationStats" --data-urlencode
'teams=[1,2]'  --data-urlencode 'from=1513508950' --data-urlencode
'to=1513595370' --data-urlencode 'aggregation=sum'
```

Successful request return:

```
[{
        "datapoints": [
                [2.0, 1513657700000],
                [9.0, 1513661300000],
                [1.0, 1513664900000],
                [4.0, 1513668500000],
                [3.0, 1513672100000],
                [2.0, 1513675700000],
                [2.0, 1513679300000],
                [9.0, 1513682900000],
                [1.0, 1513686500000]
        ],
        "target": "Cloud DevOps"
}, {
        "datapoints": [],
        "target": "Database"
}]
```

## Workflow

### addProcess

A POST request that adds a new process to the database.

Processes are external business entities related to business activities that are affected by the incidents that Cisco Crosswork Situation Manager captures in Situations.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **name** | String | The process name. |
| **description** | String | The process description. Optional. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/addProcess"
-H "Content-Type: application/json; charset=UTF-8" -d '{"name" : "new proc
1", "description" : "This is my description 12345"}'
```

Successful return:

NO RESPONSE TEXT

### addService

A POST request that adds a new external service to the database.

An external service is a business entity monitored by Cisco Crosswork Situation Manager via Event streams.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **name** | String | The service name. |
| **description** | String | The service description. Optional. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/addService"
-H "Content-Type: application/json; charset=UTF-8" -d '{"name" : "new svc
1", "description" : "This is my description 12345"}'
```

Successful return:

NO RESPONSE TEXT

### createMaintenanceWindow

A POST request that creates a maintenance window. The maintenance window filters alerts caused by a known period of maintenance.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **auth_token** | String<br><br>(Optional) | A valid **auth_token** returned from the **authenticate** request. |
| **name** | String<br><br>(Required) | Name of the maintenance window. |
| **description** | String<br><br>(Required) | Description of the maintenance window. |
| **filter** | String<br><br>(Required) | JSON or SQL-like filter for alerts to match. The filter must be in JSON format, that is, the same format used in alert and Situation filters in the database. |
| **start_date_time** | Epoch Seconds (Number)<br><br>(Required) | Start time of themaintenancewindow. This must be in epoch time and may be up to 5 years in the future. |
| **duration** | Seconds (Number)<br><br>(Required) | Duration of the maintenance window in seconds. The minimum duration is 1 second and the maximum is 157784630 seconds (5 years). |
| **forward_alerts** | Boolean<br><br>(Required) | Defines whether or not alerts should be forwarded to the next Moolet in the processing chain. |
| **recurring_period** | Number<br><br>(Optional) | Whether or not this is a recurring maintenance window. Set this to:<br><br>**1** for a recurring maintenance window.<br><br>**0** for a one-time maintenance window.<br><br>If not specified, defaults to **0**. If you set this property to **1**, you must specify **recurring_period_units**. |
| **recurring_period_units** | Number<br><br>(Optional) | Specifies the recurring period of the maintenance window, in days, weeks or months. Defaults to **0** if**recurring_period**is set to**0**.Valid values are:<br><br>**2** = daily<br><br>**3** = weekly<br><br>**4** = monthly |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object with the following:

| Name | Type | Description |
|---|---|---|
| **window_id** | Number | The new maintenance window ID. |

Example

Example cURL request to create a window, which recurs once a month (from its start_date_time), with a filterwhere thesource is 'server1' and the external ID is one of 'value1', 'value2', or 'value3':

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/createMaintenanceWindow" -H "Content-Type:
application/json; charset=UTF-8" -d '{"name":"window1",
"description":"window1 description here", "filter": "source = \"server1\"
and external_id in (\"value1\", \"value2\", \"value3\")",
"start_date_time": 1473849237, "duration": 55800, "forward_alerts": false,
"recurring_period": 1, "recurring_period_units": 4}'
```

Example cURL request to create a one-time maintenance window, which is filtered when thesource is equal to 'hostWhichIsDown':

```
curl "https://<YOUR_HOSTNAME>:8080/graze/v1/createMaintenanceWindow" -H
"Content-Type: application/json; charset=UTF-8" --insecure -X POST -v --
data '{"auth_token": "<YOUR_GRAZE_AUTH_TOKEN>", "name": "my_window_1",
"description": "This is my description", "filter": { "column": "source",
"op": 0, "value": "hostWhichIsDown", "type": "LEAF" }, "start_date_time":
1473849237, "duration": 55800, "forward_alerts": false}
```

Successful request return:

```
{"window_id":16}
```

### deleteMaintenanceWindow

A POST request that deletes a single maintenance window.

| Parameter | Type | Required | Description |
|---|---|---|---|
| **auth_token** | String | No | A valid **auth_token** returned from the **authenticate** request. |
| **id** | Number | Yes | ID of the maintenance window you want to delete. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

Example cURL request to delete maintenance window 123:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/deleteMaintenanceWindow" -H "Content-Type:
application/json; charset=UTF-8" -d '{"id"[123]}'
```

Successful return:

NO RESPONSE TEXT

### deleteMaintenanceWindows

A POST request that deletes maintenance windows that match the specified filter.

| Parameter | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **filter** | String | ID of the maintenance window to delete. |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL request to delete maintenance windows that match the filter where the ID is 3 or 4:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/deleteMaintenanceWindows" -H "Content-Type:
application/json; charset=UTF-8" -d '{"filter":"id in (3,4)"}'
```

Example cURL request to delete maintenance windows that match the filter where the host is "CSF_RD_243:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/deleteMaintenanceWindows" -H "Content-Type:
application/json; charset=UTF-8" -d '{"filter":"host matches
\"CSF_RD_243\""}'
```

Successful return:

NO RESPONSE TEXT

### findMaintenanceWindows

A GET request that returns maintenance windows that match a filter.

Request Arguments

| Parameter | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| filter | String | SQL or JSON filter. |
| limit | Number | The number of windows to fetch (defaults to 100). |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/findMaintenanceWindows" --data-urlencode
'filter=description matches "My"' --data-urlencode 'limit=2'
```

Successful request return:

```
{"windows":[{"filter":"{ \"column\": \"type\", \"op\": 10, \"value\":
\"KnownErrorType1234\", \"type\": \"LEAF\"
}","duration":3600,"recurring_period":1,"del_flag":false,"forward_alerts":
false,"last_updated":1499425460,"name":"My window
1","updated_by":5,"description":"My description
1","id":1,"recurring_period_units":3,"start_date_time":1499425427},{"filte
r":"{ \"column\": \"description\", \"op\": 10, \"value\":
\"FireInServerRoom\", \"type\": \"LEAF\"
}","duration":3600,"recurring_period":0,"del_flag":false,"forward_alerts":
false,"last_updated":1499425489,"name":"My second
window","updated_by":5,"description":"Technical details
here","id":2,"recurring_period_units":0,"start_date_time":1499425462}]}
```

### getIntegrationConfig

A GET request that exports the configuration and mapping needed for an integration in JSON format.

The exported JSON file can be saved as a duplicate configuration of the original integration. For example, you can modify and save the returned object as **webhook_lam_custom.conf**. Run it with this command:

```
$MOOGSOFT_HOME/bin/webhook_lam --config=webhook_lam_custom.conf
```

Request Arguments

| Parameter | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **integration_id** | Number | A unique identifier given to each integration by Cisco Crosswork Situation Manager. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getIntegrationConfig?integration_id=1"
```

Successful requests return aJSON object:

```
{
    "config": {
        "filter": {
            "presend": "WebhookLam.js"
        },
        "process": "webhook_lam_webhook1",
        "conversions": {
            "sevConverter": {
                "output": "INTEGER",
                "lookup": "severity",
                "input": "STRING"
            },
            "stringToInt": {
                "output": "INTEGER",
                "input": "STRING"
            }
        },
        "mapping": {
            "catchAll": "overflow",
            "rules": [
                {
                    "name": "signature",
                    "rule": "$signature"
                },
                {
                    "name": "source_id",
                    "rule": "$source_id"
                },
                {
                    "name": "external_id",
                    "rule": "$external_id"
                },
                {
                    "name": "manager",
                    "rule": "$manager"
                },
                {
                    "name": "source",
                    "rule": "$source"
                },
```

```
            {
                "name": "class",
                "rule": "$class"
            },
            {
                "name": "agent",
                "rule": "$LamInstanceName"
            },
            {
                "name": "agent_location",
                "rule": "$agent_location"
            },
            {
                "name": "type",
                "rule": "$type"
            },
            {
                "name": "severity",
                "rule": "$severity",
                "conversion": "sevConverter"
            },
            {
                "name": "description",
                "rule": "$description"
            },
            {
                "name": "agent_time",
                "rule": "$agent_time",
                "conversion": "stringToInt"
            }
        ]
    },
    "agent": {
        "name": "webhook_lam_webhook1"
    },
    "monitor": {
        "address": "localhost",
        "authentication_cache": true,
        "use_ssl": false,
        "auto_port_assign": true,
        "authentication_type": "basic",
        "rpc_response_timeout": 20,
        "lists_contain_multiple_events": true,
        "proxy": "https://freida7/events/webhook_webhook1",
        "accept_all_json": true,
        "port": 51000,
        "name": "Webhook Lam Monitor (Webhook1)",
        "num_threads": 5,
        "rest_response_mode": "on_receipt",
        "class": "CRestMonitor"
    },
    "constants": {
        "severity": {
            "0": 2,
            "moog_lookup_default": 1,
            "3": 5,
            "5": 4,
            "CLEAR": 0,
            "2": 3,
            "MAJOR": 4,
            "CRITICAL": 5,
            "MINOR": 3,
            "INDETERMINATE": 1,
```

```
                    "1": 2
                }
            }
        }
    }
```

### getMaintenanceWindows

A GET request that returns maintenance windows based on the window ID and how many should be fetched. Only returns active recurring windows and scheduled maintenance windows, not expired or deleted maintenance windows.

Request Arguments

| Parameter | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| start | Number | The start point for where to fetch windows from, that is, **0** to start at the first, **10** to start at the 11th. |
| limit | Number | The number of windows to fetch. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

Example cURL command to return the first 20 maintenance windows:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getMaintenanceWindows" --data-urlencode
'start=0' --data-urlencode 'limit=20'
```

Successful return:


```
{"windows":[{"filter":"{ \"column\": \"type\", \"op\": 10, \"value\":
\"KnownErrorType1234\", \"type\": \"LEAF\"
}","duration":3600,"recurring_period":1,"del_flag":false,"forward_alerts":
false,"last_updated":1499425460,"name":"My window
1","updated_by":5,"description":"My description
1","id":1,"recurring_period_units":3,"start_date_time":1499425427},{"filte
r":"{ \"column\": \"description\", \"op\": 10, \"value\":
\"FireInServerRoom\", \"type\": \"LEAF\"
}","duration":3600,"recurring_period":0,"del_flag":false,"forward_alerts":
false,"last_updated":1499425489,"name":"My second
window","updated_by":5,"description":"Technical details
here","id":2,"recurring_period_units":0,"start_date_time":1499425462}]}
```


### getProcesses

A GET request that returns the list of processes.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. Required. |
| **limit** | Integer | The maximum number of results that are returned. Optional. Default is 1000. |

Return Parameters

| Type | Description |
|---|---|

| | |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **process_id** | Number | The ID of the process. |
| **name** | String | The name of the process. |
| **description** | String | The description of the process. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getProcesses"
```

Successful request return:

```
[
    {
        "process_id": 1,
        "name": "Example1",
        "description": "Example1"
    },
    {
        "process_id": 2,
        "name": "Example2",
        "description": "Example2"
    },
    {
        "process_id": 3,
        "name": "Example3",
        "description": "Example3"
    }
]
```

### getServices

A GET request that returns the list of services.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. Required. |
| **limit** | Integer | The maximum number of results that are returned. Optional. Default is 1000. |

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **service_id** | Number | The ID of the service. |
| **name** | String | The service name. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getServices"
```

Successful request return:

```
[{
        "service_id": 15,
        "name": "Service A"
}, {
        "service_id": 4,
        "name": "Service B"
}]
```

### getSeverities

A GET request that returns the list of possible severities and severity IDs.

No Requested Arguments

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **severity_id** | Number | The ID of the severity. |
| **name** | String | The severity name. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getSeverities"
```

Successful request return:

```
[{
        "name": "Clear",
        "severity_id": 0
}, {
        "name": "Indeterminate",
        "severity_id": 1
}, {
        "name": "Warning",
        "severity_id": 2
}, {
        "name": "Minor",
        "severity_id": 3
}, {
        "name": "Major",
        "severity_id": 4
}, {
        "name": "Critical",
        "severity_id": 5
}]
```

### getStats

A GET request that list all endpoints available with their description.

No Requested Arguments

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getStats"
```

Successful request return:

```
[{
        "endpoint": "getTeamSituationStats",
        "description": "returns the number of active situations assign to
a team over time",
        "display_name": "Open Situations by Team",
        "parameters": {
                "teams": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getTeams",
                                "value": "team_id"
                        },
                        "type": "mapped",
                        "required": false
                },
                "from": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long",
                        "required": true
                },
                "to": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long",
                        "required": true
                }
        }
}, {
        "endpoint": "getServiceSituationStats",
        "description": "returns the number of active situations impacting
a service over time",
        "display_name": "Open Situations by Service",
        "parameters": {
                "teams": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getTeams",
                                "value": "team_id"
                        },
                        "type": "mapped",
                        "required": false
                },
                "from": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long",
                        "required": true
                },
```

```
                "services": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getServices",
                                "value": "service_id"
                        },
                        "type": "mapped",
                        "required": false
                },
                "to": {
                        "description": "A timestamp from epoch in
  seconds",
                        "type": "Long",
                        "required": true
                }
        }
  }, {
        "endpoint": "getSystemSituationStats",
        "description": "returns the number of active situations in the
  system over time",
        "display_name": "Open Situations",
        "parameters": {
                "from": {
                        "description": "A timestamp from epoch in
  seconds",
                        "type": "Long",
                        "required": true
                },
                "to": {
                        "description": "A timestamp from epoch in
  seconds",
                        "type": "Long",
                        "required": true
                }
        }
  }, {
        "endpoint": "getStatusSituationStats",
        "description": "returns the number of active situations with
  specified status over time",
        "display_name": "Situations by Status",
        "parameters": {
                "teams": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getTeams",
                                "value": "team_id"
                        },
                        "type": "mapped",
                        "required": false
                },
                "from": {
                        "description": "A timestamp from epoch in
  seconds",
                        "type": "Long",
                        "required": true
                },
                "to": {
                        "description": "A timestamp from epoch in
  seconds",
                        "type": "Long",
                        "required": true
                },
                "status": {
                        "mapping": {
```

```
                                "display_value": "name",
                                "endpoint": "getStatuses",
                                "value": "status_id"
                        },
                        "type": "mapped",
                        "required": false
                }
        }
}, {
        "endpoint": "getSeveritySituationStats",
        "description": "returns the number of active situations with
specified severity over time",
        "display_name": "Open Situations by severity",
        "parameters": {
                "severity": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getSeverities",
                                "value": "severity_id"
                        },
                        "type": "mapped",
                        "required": "false"
                },
                "teams": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getTeams",
                                "value": "team_id"
                        },
                        "type": "mapped",
                        "required": false
                },
                "from": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long",
                        "required": true
                },
                "to": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long",
                        "required": true
                }
        }
}, {
        "endpoint": "getMTTAStats",
        "description": "returns the mean time to acknowledge a situation
over time",
        "display_name": "Mean Time To Acknowledge situations",
        "parameters": {
                "teams": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getTeams",
                                "value": "team_id"
                        },
                        "type": "mapped",
                        "required": false
                },
                "from": {
                        "description": "A timestamp from epoch in
seconds",
```

```
                        "type": "Long",
                        "required": true
                },
                "to": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long",
                        "required": true
                }
        }
}, {
        "endpoint": "getMTTDStats",
        "description": "returns the mean time to detect a situation over
time",
        "display_name": "Mean Time To Detect situations",
        "parameters": {
                "teams": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getTeams",
                                "value": "team_id"
                        },
                        "type": "mapped",
                        "required": false
                },
                "from": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long",
                        "required": true
                },
                "to": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long",
                        "required": true
                }
        }
}, {
        "endpoint": "getMTTRStats",
        "description": "returns the mean time to resolve a situation over
time",
        "display_name": "Mean Time To Resolve situations",
        "parameters": {
                "teams": {
                        "mapping": {
                                "display_value": "name",
                                "endpoint": "getTeams",
                                "value": "team_id"
                        },
                        "type": "mapped",
                        "required": false
                },
                "from": {
                        "description": "A timestamp from epoch in
seconds",
                        "type": "Long ",
                        "required ": true
                },
                "to ": {
                        "description ": "A timestamp from epoch in seconds
",
                        "type ": "Long ",
                        "required ": true
```

```
                }
        }
}, {
        "endpoint ": "getReassignedSituationStats ",
        "description ": "returns the number of situations that have been
reassigned over time ",
        "display_name ": "Reassigned situations ",
        "parameters ": {
                "teams ": {
                        "mapping ": {
                                "display_value ": "name ",
                                "endpoint ": "getTeams ",
                                "value ": "team_id "
                        },
                        "type ": "mapped ",
                        "required ": false
                },
                "from ": {
                        "description ": "A timestamp from epoch in seconds
",
                        "type ": "Long ",
                        "required ": true
                },
                "to ": {
                        "description ": "A timestamp from epoch in seconds
",
                        "type ": "Long ",
                        "required ": true
                }
        }
}]
```

### getStatuses

A GET request that returns a list of statuses that can apply to Situations and their IDs.

No Requested Arguments

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return an array of JSON objects containing the following:

| Name | Type | Description |
|---|---|---|
| **status_id** | Number | The ID of the status. |
| **name** | String | The status name. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getStatuses"
```

Successful request return:

```
[{
        "status_id": 1,
        "name": "Opened"
}, {
        "status_id": 2,
```

```
            "name": "Unassigned"
    }, {
            "status_id": 3,
            "name": "Assigned"
    }, {
            "status_id": 4,
            "name": "Acknowledged"
    }, {
            "status_id": 5,
            "name": "Unacknowledged"
    }, {
            "status_id": 6,
            "name": "Active"
    }, {
            "status_id": 7,
            "name": "Dormant"
    }, {
            "status_id": 8,
            "name": "Resolved"
    }, {
            "status_id": 9,
            "name": "Closed"
    }, {
            "status_id": 10,
            "name": "SLA Exceeded"
    }]
```

### getSystemStatus

A GET request that returns current system status information for all processes.

Request Argument

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |

There are no other arguments, as this method returns data on all processes.

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **component** | String | Represents the name of a component within the process. May not be present, depending on the process. |
| **instance** | String | The instance name. |
| **last_heartbeat** | Number | The timestamp (in Unix epoch time) of the last process heartbeat. 0 is a special value indicating that a heartbeat has never been received. |
| **missed_heartbeats** | Number | The number of missed process heartbeats. -1 is a special value indicating that a heartbeat has never been received. |
| **process_name** | String | The process name. |
| **processes** | List | A list of the processes, with status information. |
| **reserved** | Boolean | Indicates whether the process is reserved: |

| | | |
|---|---|---|
| | | **true** = a reserved process<br>**false** = process not reserved<br>A reserved process is a process that is usually required for Cisco Crosswork Situation Manager to be working properly. |
| **running** | Boolean | Indicates whether the process is running or not:<br>**true** = running<br>**false** = not running |
| **service_name** | String | The service name. |
| **display_name** | String | The name of the service in the configuration. |
| **type** | String | The type of the service, for example, lam, servlets, moog_farmd. |
| **passive** | Boolean | Indicates whether the service is passive in a HA environment:<br>**true** = passive<br>**false** = active |
| **stoppable** | Boolean | Indicates whether or not the service can be stopped:<br>**true** = stoppable<br>**false** = not stoppable |
| **ha_conf** | JSON Object | A Json blob containing the HA configuration. |
| **additional_health_info** | JSON Object | Additional health information. The **pools** section includes health information for processes with an internal pool. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getSystemStatus"
```

Successful request return:


```
{
    "processes": [{
        "running": true,
        "sub_components": {
        "moogpoller": {
            "run_on_startup": true,
            "instance": "",
            "service_name": "apache-tomcat",
            "display_name": "servlets",
            "type": "servlets",
            "last_heartbeat": 1491385834300,
            "passive": false,
            "running": true,
            "component": "moogpoller",
            "reserved": true,
            "stoppable": true,
            "missed_heartbeats": 0,
            "ha_conf": {
                "cluster": "MOO",
                "instance": "",
```

```
                "default_leader": true,
                "start_as_passive": false,
                "only_leader_active": false,
                "group": "servlets"
              }
           },
        "moogsvr": {
            "run_on_startup": true,
            "instance": "",
            "service_name": "apache-tomcat",
            "display_name": "servlets",
            "type": "servlets",
            "last_heartbeat": 1491385825246,
            "passive": false,
            "running": true,
            "component": "moogsvr",
            "reserved": true,
            "stoppable": true,
            "missed_heartbeats": 0,
            "ha_conf": {
                "cluster": "MOO",
                "instance": "",
                "default_leader": true,
                "start_as_passive": false,
                "only_leader_active": false,
                "group": "servlets"
            }
         }
      },
    "instance": "",
        "reserved": true,
        "service_name": "apache-tomcat",
        "stoppable": true,
        "missed_heartbeats": 0,
        "display_name": "servlets",
        "type": "servlets",
        "last_heartbeat": 1491385834300,
        "ha_conf": {
            "cluster": "MOO",
            "instance": "",
            "default_leader": true,
            "start_as_passive": false,
            "only_leader_active": false,
            "group": "servlets"
            },
        "passive": false
            }, {
                "running": false,
                "instance": "",
                "last_missed_heartbeat": 1491385820601,
                "reserved": false,
                "stoppable": false,
                "missed_heartbeats": 10,
                "display_name": "test_lam",
                "type": "lam",
                "last_heartbeat": 1491382820601,
                "additional_health_info": {
                        "thread_pool_queue_size": 0,
                        "published_events": {
                                "last_5_minutes": 130,
                                "last_10_minutes": 130,
                                "last_minute": 130
                        }
                },
```

```
                "ha_conf": {
                        "cluster": "MOO",
                        "instance": "",
                        "default_leader": true,
                        "start_as_passive": false,
                        "only_leader_active": true,
                        "group": "test_lam"
                },
                "passive": false
        "sub_components": {
            "SituationMgr": {
            "run_on_startup": true,
            "instance": "",
            "last_missed_heartbeat": 1491385821669,
            "service_name": "moogfarmd",
            "display_name": "moog_farmd",
            "type": "moog_farmd",
            "last_heartbeat": 1491382821669,
            "passive": false,
            "running": false,
            "component": "SituationMgr",
            "reserved": true,
            "stoppable": true,
            "missed_heartbeats": 10,
            "ha_conf": {
                "cluster": "MOO",
                "instance": "",
                "default_leader": true,
                "start_as_passive": false,
                "only_leader_active": true,
                "group": "moog_farmd"
             }
          },
    "AlertBuilder": {
        "run_on_startup": true,
        "instance": "",
        "last_missed_heartbeat": 1491385821669,
        "service_name": "moogfarmd",
        "display_name": "moog_farmd",
        "type": "moog_farmd",
        "last_heartbeat": 1491382821669,
        "passive": false,
        "running": false,
        "component": "AlertBuilder",
        "reserved": true,
        "stoppable": true,
        "missed_heartbeats": 10,
        "ha_conf": {
            "cluster": "MOO",
            "instance": "",
            "default_leader": true,
            "start_as_passive": false,
            "only_leader_active": true,
            "group": "moog_farmd"
         }
       },
    "TeamsMgr": {
        "run_on_startup": true,
        "instance": "",
        "last_missed_heartbeat": 1491385821669,
        "service_name": "moogfarmd",
        "display_name": "moog_farmd",
        "type": "moog_farmd",
```

```
            "last_heartbeat": 1491382821669,
            "passive": false,
            "running": false,
            "component": "TeamsMgr",
            "reserved": true,
            "stoppable": true,
            "missed_heartbeats": 10,
            "ha_conf": {
                "cluster": "MOO",
                "instance": "",
                "default_leader": true,
                "start_as_passive": false,
                "only_leader_active": true,
                "group": "moog_farmd"
             }
          }
        },
        "instance": "",
        "last_missed_heartbeat": 1491385821669,
        "service_name": "moogfarmd",
        "display_name": "moog_farmd",
        "type": "moog_farmd",
        "last_heartbeat": 1491382821669,
        "additional_health_info": {
            "event_processing_metric": 0.65
        },
    "passive": false,
    "running": false,
    "reserved": true,
    "stoppable": true,
    "missed_heartbeats": 10,
    "ha_conf": {
        "cluster": "MOO",
        "instance": "",
        "default_leader": true,
        "start_as_passive": false,
        "only_leader_active": true,
        "group": "moog_farmd"
        }
     },
    {
        "running": false,
        "instance": "",
        "reserved": false,
        "service_name": "restclientlamd",
        "stoppable": true,
        "display_name": "rest_client_lam",
        "type": "lam",
        "ha_conf": {
            "cluster": "MOO",
            "instance": "",
            "group": "rest_client_lam"
         }
         "additional_health_info": {
             "pools": {
                 "MoogPoller": [{
                 "removed": 0,
                 "ration": 0.0,
                 "busy": 0,
                 "resource_type": "com.mysql.jdbc.JDBC4Connection",
                 "checkout_per_second": 0.0,
                 "free": 10,
                 "avg_checkedout_seconds": 0.0,
                 "capacity": 10
```

```
            }],
            "Message sender pool": [{
                "removed": 0,
                "ration": 0.0,
                "busy": 0,
                "resource_type": "com.moogsoft.mooms.CMoomsMessageSender",
                "checkout_per_second": 0.09997000899730081,
                "free": 10,
                "avg_checkedout_seconds": 0.002,
                "capacity": 10
              }]
        }
    }]
}
```


### getSystemSummary

A GET request that returns a summary of current alerts and Situations in Cisco Crosswork Situation Manager.

Request Argument

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |

There are no other arguments, as this method returns data on all alerts and Situations.

Return Parameters

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object **system_summary**, containing Cisco Crosswork Situation Manager statistics in the following:

| Name | Type | Description |
|---|---|---|
| **open_sitns** | Number | The number of open Situations in Cisco Crosswork Situation Manager. |
| **open_sitns_down** | Number | The number of open Situations that are trending down. |
| **open_sitns_up** | Number | The number of open Situations that are trending up. |
| **avg_events_per_sitn** | Number | The average number of events per situation. |
| **avg_alerts_per_sitn** | Number | The average number of alerts per situation. |
| **service_count** | Number | The number of services currently in Cisco Crosswork Situation Manager. |
| **open_sigs_unassigned** | Number | The number of situations unassigned. |
| **total_events** | Number | The total number of Events currently in Cisco Crosswork Situation Manager. |

Example

cURL command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getSystemSummary"
```

Successful request return:


```
{"system_summary":{"total_events":61676,"open_sitns":571,"avg_events_per_s
itn":305,"open_sitns_up":565,"open_sitns_down":2,"avg_alerts_per_sitn":16,
"open_sigs_unassigned":310,"timestamp":1499425056}}
```

**updateMaintenanceWindow**

A POST request that updates an existing maintenance window.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **window_id** | String<br><br>(Required) | ID of the existing maintenance window. |
| **auth_token** | String<br><br>(Optional) | A valid **auth_token** returned from the **authenticate** request. |
| **name** | String<br><br>(Optional) | Name of the maintenance window. |
| **description** | String<br><br>(Optional) | Description of the maintenance window. |
| **filter** | String<br><br>(Optional) | JSON or SQL-like filter for alerts to match. The filter must be in JSON format, that is, the same format used in alert and Situation filters in the database. You cannot change the filter of an active maintenance window. |
| **start_date_time** | Epoch Seconds (Number)<br><br>(Optional) | Start time of the maintenance window. This must be in epoch time and may be up to 5 years in the future. You cannot change the **start_date_time** of an active maintenance window. |
| **duration** | Seconds (Number)<br><br>(Optional) | Duration of the maintenance window in seconds. The minimum duration is 1 second and the maximum is 157784630 seconds (5 years). |
| **forward_alerts** | Boolean<br><br>(Optional) | Whether or not alerts are forwarded to the next Moolet in the processing chain. If you change this from **false** to **true**, only alerts received after the change are forwarded. |
| **recurring_period** | Number<br><br>(Optional) | Whether or not this is a recurring maintenance window. Set this to:<br><br>**1** for a recurring maintenance window.<br><br>**0** for a one-time maintenance window.<br><br>If you change this from **0** to **1**, you must specify **recurring_period_units**. |
| **recurring_period_units** | Number<br><br>(Optional) | Specifies the recurring period of the maintenance window, in days, weeks or months. If you set **recurring_period** to **0**, you must set **recurring_period_units** to **0**. Valid values are:<br><br>**0** = a one-time maintenance window<br><br>**2** = daily<br><br>**3** = weekly<br><br>**4** = monthly |

Return Parameter

| Type | Description |
|---|---|

| | |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Examples

Example cURL request to update all the parameters in the existing maintenance window 351:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/updateMaintenanceWindow" -H "Content-Type:
application/json; charset=UTF-8" -d '{"window_id":351, "name":"Updated
name", "description":"Updated Description", "filter":"source =
\"server1\"", "start_date_time":1546433400, "duration":3600,
"forward_alerts":false, "recurring_period":1, "recurring_period_units":3}'
```

Example cURL request to update the existing maintenance window 27 so that it will not occur again:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/updateMaintenanceWindow" -H "Content-Type:
application/json; charset=UTF-8" -d '{"window_id":27,
"recurring_period":0, "recurring_period_units":0}'
```

Successful return:

A successful request returns an HTTP 200 response.

### POST Parameters

You can send POST parameters as **`form-urlencoded`** or as **`application/json`** parameters.

### form-urlencoded

POST parameters can be sent as **`form-urlencoded`** parameters. To do this, the content type must be set to **`application/x-www-form-urlencoded`**. If the character set is not set, then UTF-8 is assumed.

cURL command:

```
"https://localhost/graze/v1/resolveSituation?auth_token=b40244fd79aa46fba7
6c60c56d538c49&sitn_id=10" --insecure -X POST -v
```

### application/json

POST parameters can be supplied as JSON within the body of the request. To do this, the content type must be set to **`application/json`**. If the character set is not set, then UTF-8 is assumed.

cURL command:

```
"https://localhost/graze/v1/resolveSituation" -H "Content-Type:
application/json; charset=UTF-8" --insecure -X POST -v --data
'{"auth_token" : "b40244fd79aa46fba76c60c56d538c49","sitn_id" : 10}
```

## 'HTTP Status and Error Codes

The Graze API returns the following HTTP status and error codes for successful and unsuccessful requests:

| HTTP Code | Meaning |
|---|---|
| 200 | Successful request. |
| 400 | Incorrectly formatted request. |
| 401 | A request with an invalid or expired **`auth_code`**. |
| 403 | Forbidden request. |
| 404 | Not found, for example, the **`sitn_id`** could not be found because it does not exist. |

| | |
|---|---|
| 500 | Failed request, for example, due to an invalid **`sitn_id`**. |

## Situation Action Codes

The getSituationActions Graze API endpoint and MoogDb V2 method can retrieve actions that happened on a given Situation.

The table below shows the list of IDs and the matching description for each action:

| Event Id | Description |
|---|---|
| 1 | Situation Created |
| 2 | Assigned Moderator |
| 3 | Situation Resolved |
| 4 | Situation Revived |
| 5 | Situation Closed |
| 6 | Assigned Queue |
| 7 | Created By Merge |
| 8 | Used In Merge |
| 9 | Created By Split |
| 10 | Used For Split |
| 11 | Ran Tool |
| 12 | Acknowledged Situation Moderator |
| 13 | Deacknowledged Situation Moderator |
| 14 | Added Alerts To Situation |
| 15 | Added Entry To Thread |
| 16 | Changed Situation Processes |
| 17 | Changed Situation Services |
| 18 | Created Thread |
| 19 | Agreed With Thread Entry |
| 21 | Commented On Thread Entry |
| 22 | Disagreed With Thread Entry |
| 23 | Changed Situation Custom Info |
| 24 | Described Situation |
| 25 | Excluded User |
| 26 | Invited User |
| 27 | Moved Alerts To Situation |
| 28 | Removed Alerts From Situation |
| 29 | Situation Updated |

| | |
|---|---|
| 30 | Situation Teams Changes |
| 31 | Marked Thread Entry As Resolving |
| 32 | Unmarked Thread Entry As Resolving |
| 33 | Situation Rated |
| 34 | Situation Rating Removed |
| 35 | Situation Internal Severity Changed |
| 36 | Situation Superseding Others |
| 37 | Updated Comment On Thread Entry |
| 38 | Updated Entry Of Thread |

## Alert Action Codes

The getAlertActions Graze API endpoint and the MoogDb V2 method can retrieve actions that happened on a given alert.

The table below shows the list of IDs and the matching description for each action:

| Event Id | Description |
|---|---|
| 0 | Alert Created |
| 2 | Event Added to Alert |
| 3 | Alert Assigned |
| 4 | Alert Updated |
| 5 | Alert Updated Custom Info |
| 6 | Alert Added to Situation |
| 7 | Team Updated |
| 8 | Alert Resolved |
| 9 | Alert Closed |
| 10 | Ran Tool |

## Stats API

You can use the Stats API endpoints to report on Cisco Crosswork Situation Manager data.

These endpoints return various statistics about teams, Situations and services. You can also fetch information on the Mean Time to Acknowledge (MTTA), Mean Time to Detect (MTTD) and Mean Time to Resolve (MTTR).

### System Endpoints

The following endpoints return data statistics relating to your Cisco Crosswork Situation Manager system:

#### getAlertsInNewSituationsStats

A GET request that returns the number of alerts that belong to new Situations during a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | "accumulate" - gradually adds data points together over time.<br><br>"none" - no aggregation of data points. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Alerts in new situations" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [datapoint, timestamp]:<br><br>Data point: Number of alerts.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of alerts each hour in the time period.<br><br>1 week to 1 month: Returns the number of alerts each day in the time period.<br><br>1 month to 1 year: Returns the number of alerts each week in the time period.<br><br>More than 1 year Returns the number of alerts each month in the time period. |

Request Example

A cURL command that requests all alerts in new Situations over a 24 hour time range from 13.23pm on Tuesday 18th September until 13:24pm on Wednesday 19th September 2018:

```
curl -G -u graze:graze -k -v
"https://freida7/graze/v1/getAlertsInNewSituationsStats" --data-urlencode
'from=1537277017' --data-urlencode 'to=1537363453'
```

Response Example

A successful response indicating there were 56 alerts at13:23pm on Wednesday 19th September 2018:

```
[
        {"datapoints":[
        [56.0,1537359817000]
        ],
        "target":"Alerts in new situations"}
]
```

### getMTTAStats

A GET request that returns the Mean Time To Acknowledge (MTTA) Situations in a system over time.

The time to acknowledge (TTA) for a Situation is the duration from the first event's inclusion in the Situation to the time when a moderator assigns a Situation to a user in Cisco Crosswork Situation Manager.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error codeindicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Mean Time to Acknowledge (MTTA)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: MTTA (seconds) for that bucket.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of the data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the MTTA each hour in the time period.<br><br>1 week to 1 month: Returns the MTTA each day in the time period.<br><br>1 month to 1 year: Returns the MTTA each week in the time period.<br><br>More than 1 year: Returns the MTTA each month in the time period. |

Request Example

A cURL command to return the MTTA for Cisco Crosswork Situation Manager over a 24 hour time range from 11.09am on Sunday 17th December until 11.09am on Monday 18th December 2017:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getMTTAStats" --
data-urlencode 'from=1513508950' --data-urlencode 'to=1513595370'
```

Response Example

A successful response returns the MTTA in seconds for each hour:

```
[{
        "datapoints": [
                [312.0, 1513657700000],
                [209.0, 1513661300000],
                [101.0, 1513664900000],
                [114.0, 1513668500000],
                [203.0, 1513672100000],
                [120.0, 1513675700000],
                [201.0, 1513679300000],
                [90.0, 1513682900000],
                [100.0, 1513686500000]
        ],
        "target": "Mean Time to Acknowledge (MTTA)"
}]
```

### getMTTDStats

A GET request that returns the Mean Time To Detect (MTTD) Situations in the system over time. The time to detect (TTD) for a Situation is the duration from the first event's inclusion in the Situation to the Situation creation time.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error codeindicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Mean Time to Detect (MTTD)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: MTTD (seconds) for that bucket.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of the data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the MTTD each hour in the time period.<br><br>1 week to 1 month: Returns the MTTD each day in the time period.<br><br>1 month to 1 year: Returns the MTTD each week in the time period.<br><br>More than 1 year: Returns the MTTD each month in the time period. |

Request Example

A cURL request to retrieve the MTTD for Cisco Crosswork Situation Manager from 11.09am on Sunday 17th December until 11.09am on Sunday 24th December 2017:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getMTTDStats" --
data-urlencode 'from=1513508950' --data-urlencode 'to=1514113750'
```

Response Example

Successful request returns the MTTD for the 24 hour time frame:

```
[{
        "datapoints": [
                [272.0, 1514113750000],
        ],
        "target": "Mean Time to Detect (MTTD)"
}]
```

### getMTTRStats

A GET request that returns the Mean Time To Resolve (MTTR) for Situations in the system over a given range of time.The TTR for a Situation is the duration from the first event in the Situation to the time when a user resolved the Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last data point returned is the current state data point. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error codeindicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Mean Time to Resolve (MTTR)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [datapoint, timestamp]:<br><br>Data point: MTTR (seconds) for that bucket.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the MTTR each hour in the time period.<br><br>1 week to 1 month: Returns the MTTR each day in the time period.<br><br>1 month to 1 year: Returns the MTTR each week in the time period.<br><br>More than 1 year: Returns the MTTR each month in the time period. |

Request Example

A cURL request to retrieve the MTTR for Cisco Crosswork Situation Manager from 11.30am on Sunday, September 24th 2017 until 11.30am on Sunday, September24th 2018:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getMTTRStats" --data-urlencode 'from=1506252610' --data-urlencode 'to=1537788610'
```

Response Example

A successful response indicates the MTTR for the year was 2.72 minutes:

```
[{
        "datapoints": [
                [163.54,1537784877233]
        ],
        "target":"Mean Time to Resolve (MTTR)"
}]
```

### getNewAlertsStats

A GET request that returns the number of new alerts over a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |

| | | |
|---|---|---|
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | "accumulate" - gradually adds data points together over time.<br><br>"none" - no aggregation of data points. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "New Alerts" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of alerts.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of alerts each hour in the time period.<br><br>1 week to 1 month: Returns the number of alerts each day in the time period.<br><br>1 month to 1 year: Returns the number of alerts each week in the time period.<br><br>More than 1 year: Returns the number of alerts each month in the time period. |

Request Example

A cURL request to retrieve the number of new alerts between Wednesday, January 17th and Thursday, January 18th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getNewAlertsStats" --data-urlencode
'from=1516216020' --data-urlencode 'to=1516282420'
```

Response Example

Successful response indicates there were 28,542 new alerts over the 24 hour time period:

```
[
        {"datapoints":[
                [28542.0,1523438216685]
        ],
        "target":"New Alerts"}
]
```

### getNewAlertsPerSituationsStats

A GET request that returns the percentage alert to Situation noise reduction for a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |

| | | |
|---|---|---|
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "New Alerts per Situation" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Percentage noise reduction (alert to Situation reduction).<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the percentage noise reduction each hour in the time period.<br><br>1 week to 1 month: Returns the percentage noise reduction each day in the time period.<br><br>1 month to 1 year: Returns the percentage noise reduction each week in the time period.<br><br>More than 1 year: Returns the percentage noise reduction each month in the time period. |

Request Example

A cURL request to retrieve the percentage noise reduction from 7.07pm on Wednesday, 17th January 2018 until 1.33pm on Thursday, 18th January 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getNewAlertsPerSituationsStats" --data-
urlencode 'from=1516216020' --data-urlencode 'to=1516282420'
```

Response Example

A successful response indicating a noise reduction of 78.5% in the number of alerts to Situations:

```
[
        {"datapoints":[
                [78.5,1523438216685]
        ],
        "target":"New Alerts per Situation"}
]
```

### getNewEventsPerAlertsStats

A GET request that returns the percentage event to alert noise reduction over a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds. |

| | | |
|---|---|---|
| | | If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "New Events per Alerts" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Percentage noise reduction (event to alert reduction).<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the percentage noise reduction each hour in the time period.<br><br>1 week to 1 month: Returns the percentage noise reduction each day in the time period.<br><br>1 month to 1 year: Returns the percentage noise reduction each week in the time period.<br><br>More than 1 year: Returns the percentage noise reduction each month in the time period. |

Request Example

A cURL request that retrieves that event to alert noise reduction in Cisco Crosswork Situation Manager from 7.07pm on Wednesday, 17th January until 7.07pm on Thursday, 18th January 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getNewEventsPerAlertsStats" --data-urlencode
'from=1516216020' --data-urlencode 'to=1516302431'
```

Response Example

A successful response indicating a 58% noise reduction:

```
[
        {"datapoints":[
                [58.0,1523438216685]
        ],
        "target":"New Events per Alerts"}
]
```

### getNewEventsPerSituationsStats

A GET request that returns the percentage event to Situation noise reduction over a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "New Events per Situation" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Percentage noise reduction (event to Situation reduction).<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the percentage noise reduction each hour in the time period.<br><br>1 week to 1 month: Returns the percentage noise reduction each day in the time period.<br><br>1 month to 1 year: Returns the percentage noise reduction each week in the time period.<br><br>More than 1 year: Returns the percentage noise reduction each month in the time period. |

Request Example

A cURL request that retrieves the percentage noise reduction for the past month ranging from 10.28am on Sunday, August 26th until 10.28am on Wednesday, September 26th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getNewEventsPerSituationsStats" --data-
urlencode 'from=1533103200' --data-urlencode 'to=1535695200'
```

Response Example

A successful responses returns an 95% to 96% reduction in events to Situations for each week over the past month:

```
[
        {
        "datapoints":[
                [95.86151338591529,1535279280000],
                [95.79150698161867,1535884080000],
                [95.62050414072417,1536488880000],
                [96.08938014241262,1537093680000],
                [95.96508799542137,1537698480000]
        ],
        "target":"New Events per Situation"
        }
]
```

### getNewSituationsStats

A GET request that returns the number of new Situations created over a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |

| | | |
|---|---|---|
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "New Situations" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of new Situations.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of new Situations over a week from 6am on Saturday, September 1st until 6am on Saturday, September 8th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getNewSituationsStats" --data-urlencode
'from=1535781600' --data-urlencode 'to=1536386400'
```

Response Example

A successful response returns the number of new Situations for each day during the week range:

```
[
        {"datapoints":[
                [601.0,1535781600000],
                [523.0,1535868000000],
                [597.0,1535954400000],
                [618.0,1536040800000],
                [535.0,1536127200000],
                [628.0,1536213600000],
                [618.0,1536300000000]
        ],
        "target":"New situations"}
]
```

### getReassignedSituationStats

A GET request that returns the number of Situations reassigned in the system over a given range of time. A reassigned Situation is a Situation that a user has assigned to another user at least twice.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Reassigned Situation" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of reassigned Situations.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of reassigned Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of reassigned Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of reassigned Situations each week in the time period.<br><br>More than 1 year: Returns the number of reassigned Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of reassigned Situations over a month from 6am on Wednesday, August 1st until 6am on Saturday, September 1st 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getReassignedSituationStats" --data-urlencode
'from=1533103200' --data-urlencode 'to=1535781600'
```

Response Example

A successful response returns the number of reassigned Situations for each week during the month:

```
[{
        "datapoints": [
                [25.125,1533103200000],
                [24.1369,1533708000000],
                [25.9405,1534312800000],
                [24.8512,1534917600000],
                [25.1071,1535522400000],
        ],
        "target": "Reassigned Situation"
}]
```

### getReoccurringSituationStats

A GET request that returns the percentage of reoccurring situations in the system over a given time range.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Reoccurring Situations" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of reoccurring Situations.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of reoccurring Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of reoccurring Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of reoccurring Situations each week in the time period.<br><br>More than 1 year: Returns the number of reoccurring Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of reoccurring Situations from 6pm on Sunday, September 10th 2017 until 6pm on Monday, September 10th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getReoccurringSituationStats" --data-urlencode
'from=1505066400' --data-urlencode 'to=1536602400'
```

Response Example

A successful response returns that there were 186 reoccurring Situations during the year:

```
[{
        "datapoints": [
                [186.0, 1537980650126],
        ],
        "target": "Reoccurring situations"
}]
```

### getServiceSituationStats

A GET request that returns the number of active Situations impacting a service over a given time range.

Request Argument

| Name | Type | Description |
|---|---|---|
| **services** | Array | An array of services IDs. This is optional. If no services are provided, the endpoint does not return any data. |
| **from** | Number | Start of the time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | "sum" - for an aggregation of all teams provided.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | Service name(s). |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations impacting services.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of Situations impacting the Commerce/Compute service between 12pm and 6pm on Friday, August 10th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getServiceSituationStats" --data-urlencode
'services=[1,2]' --data-urlencode 'from=1533902400' --data-urlencode
'to=1533924000' --data-urlencode 'aggregation=sum'
```

Response Example

A successful response returns six data points for each hour during the six hour time range:

```
[{
        "datapoints": [
                [95.0,1533902400000],
```

```
                [85.0,1533906000000],
                [47.0,1533909600000],
                [7.0,1533913200000],
                [33.0,1533916800000],
                [66.0,1533920400000]
        ],
        "target":"Commerce/Compute"
}]
```

### getSeveritySituationStats

A GET request that returns the number of Situations by severity over a given time range.

Request Argument

| Name | Type | Description |
|---|---|---|
| **severity** | Array | An array of severity IDs. This is optional. If not given, it returns the default set of severities: Clear, Indeterminate, Warning, Minor, Major, Critical. |
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last data point returned is the current state datapoint. |
| **aggregation** | String | "sum" - for an aggregation of all teams provided.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the status. |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations per severity.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Input Example

A cURL request to retrieve the sum of the major and critical Situations between 12pm on Thursday, August 9th and 12pm on Friday, August 10th 2018:

```
curl -G -u graze:graze -k -v
"https://daffy.moogsoft.com/graze/v1/getSeveritySituationStats" --data-
urlencode 'from=1533816000' --data-urlencode 'to=1533902400' --data-
urlencode 'severity=[5, 4]' --data-urlencode 'aggregation=sum'
```

Output Example

A successful response returns 24 data points, one for each hour over the 24 hour range:


```
[{
        "datapoints":[
                [51.0,1533816000000],
                [44.0,1533819600000],
                [88.0,1533823200000],
                [84.0,1533826800000],
                [25.0,1533830400000],
                [34.0,1533834000000],
                [82.0,1533837600000],
                [58.0,1533841200000],
                [61.0,1533844800000],
                [52.0,1533848400000],
                [15.0,1533852000000],
                [50.0,1533855600000],
                [54.0,1533859200000],
                [50.0,1533862800000],
                [81.0,1533866400000],
                [78.0,1533870000000],
                [84.0,1533873600000],
                [28.0,1533877200000],
                [54.0,1533880800000],
                [36.0,1533884400000],
                [44.0,1533888000000],
                [47.0,1533891600000],
                [60.0,1533895200000],
                [54.0,1533898800000]],
        "target":"Critical/Major"
}]
```


### getStats

A GET request that retrieves all available Stats API endpoints along with their description and request parameters.

Request Arguments

None required.

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Request Example

A cURL request to return all available Stats API endpoints:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getStats"
```

Response Example

A successful response with all of the endpoints, descriptions and associated parameters:


```
[
 {
   "endpoint":"getTeamSituationStats",
   "description":"returns the number of active situations assign to a team
over time",
   "display_name":"Open Situations by Team",
   "parameters":{
      "teams":{
```

```
            "mapping":{
               "display_value":"name",
               "endpoint":"getTeams",
               "value":"team_id"
            },
            "type":"mapped",
            "required":false
         },
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Sum",
                  "value":"sum"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getTopTeamSituationStats",
      "description":"returns the number of active situations assign to a top
   team over time",
      "display_name":"Open Situations by Top Team",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Sum",
                  "value":"sum"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
```

```
            "required":true
         }
      }
   },
   {
      "endpoint":"getServiceSituationStats",
      "description":"returns the number of active situations impacting a
   service over time",
      "display_name":"Open Situations by Service",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Sum",
                  "value":"sum"
               }
            ],
            "required":false
         },
         "services":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getServices",
               "value":"service_id"
            },
            "type":"mapped",
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getTopServiceSituationStats",
      "description":"returns the number of active situations impacting a top
   service over time",
      "display_name":"Open Situations by Top Service",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
```

```
                "value":"none"
             },
             {
                "display_value":"Sum",
                "value":"sum"
             }
          ],
          "required":false
       },
       "to":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
       }
    }
 },
 {
    "endpoint":"getSystemSituationStats",
    "description":"returns the number of active situations in the system
 over time",
    "display_name":"All Open Situations",
    "parameters":{
       "from":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
       },
       "to":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
       }
    }
 },
 {
    "endpoint":"getStatusSituationStats",
    "description":"returns the number of active situations with specified
 status over time",
    "display_name":"Open Situations by Status",
    "parameters":{
       "from":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
       },
       "aggregation":{
          "default":"none",
          "type":"string",
          "static_mapping":[
             {
                "display_value":"None",
                "value":"none"
             },
             {
                "display_value":"Sum",
                "value":"sum"
             }
          ],
          "required":false
       },
       "to":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
```

```
        },
        "status":{
           "mapping":{
              "display_value":"name",
              "endpoint":"getStatuses",
              "value":"status_id"
           },
           "type":"mapped",
           "required":false
        }
     }
  },
  {
     "endpoint":"getSeveritySituationStats",
     "description":"returns the number of active situations with specified
  severity over time",
     "display_name":"Open Situations by Severity",
     "parameters":{
        "severity":{
           "mapping":{
              "display_value":"name",
              "endpoint":"getSeverities",
              "value":"severity_id"
           },
           "type":"mapped",
           "required":"false"
        },
        "from":{
           "description":"A timestamp from epoch in seconds",
           "type":"Long",
           "required":true
        },
        "aggregation":{
           "default":"none",
           "type":"string",
           "static_mapping":[
              {
                 "display_value":"None",
                 "value":"none"
              },
              {
                 "display_value":"Sum",
                 "value":"sum"
              }
           ],
           "required":false
        },
        "to":{
           "description":"A timestamp from epoch in seconds",
           "type":"Long",
           "required":true
        }
     }
  },
  {
     "endpoint":"getReoccurringSituationStats",
     "description":"returns the percentage of reoccurring situations in the
  system",
     "display_name":"Reoccurring situations",
     "parameters":{
        "from":{
           "description":"A timestamp from epoch in seconds",
           "type":"Long",
```

```
            "required":true
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getMTTAStats",
      "description":"returns the mean time to acknowledge a situation over
   time",
      "display_name":"Mean Time To Acknowledge",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getMTTDStats",
      "description":"returns the mean time to detect a situation over time",
      "display_name":"Mean Time To Detect",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getMTTRStats",
      "description":"returns the mean time to resolve a situation over time",
      "display_name":"Mean Time To Resolve",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getReassignedSituationStats",
      "description":"returns the number of situations that have been
```

```
reassigned over time",
   "display_name":"Reassigned Situations",
   "parameters":{
      "from":{
         "description":"A timestamp from epoch in seconds",
         "type":"Long",
         "required":true
      },
      "to":{
         "description":"A timestamp from epoch in seconds",
         "type":"Long",
         "required":true
      }
   }
},
{
   "endpoint":"getNewSituationsStats",
   "description":"returns the number of new situations over time",
   "display_name":"New Situations",
   "parameters":{
      "from":{
         "description":"A timestamp from epoch in seconds",
         "type":"Long",
         "required":true
      },
      "aggregation":{
         "default":"none",
         "type":"string",
         "static_mapping":[
            {
               "display_value":"None",
               "value":"none"
            },
            {
               "display_value":"Accumulate",
               "value":"accumulate"
            }
         ],
         "required":false
      },
      "to":{
         "description":"A timestamp from epoch in seconds",
         "type":"Long",
         "required":true
      }
   }
},
{
   "endpoint":"getNewAlertsStats",
   "description":"returns the number of new alerts over time",
   "display_name":"New Alerts",
   "parameters":{
      "from":{
         "description":"A timestamp from epoch in seconds",
         "type":"Long",
         "required":true
      },
      "aggregation":{
         "default":"none",
         "type":"string",
         "static_mapping":[
            {
               "display_value":"None",
```

```
                "value":"none"
             },
             {
                "display_value":"Accumulate",
                "value":"accumulate"
             }
          ],
          "required":false
       },
       "to":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
       }
    }
 },
 {
    "endpoint":"getNewEventsStats",
    "description":"returns the number of new events over time",
    "display_name":"New Events",
    "parameters":{
       "from":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
       },
       "aggregation":{
          "default":"none",
          "type":"string",
          "static_mapping":[
             {
                "display_value":"None",
                "value":"none"
             },
             {
                "display_value":"Accumulate",
                "value":"accumulate"
             }
          ],
          "required":false
       },
       "to":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
       }
    }
 },
 {
    "endpoint":"getAlertsInNewSituationsStats",
    "description":"returns the number of alerts in new situations over
time",
    "display_name":"Alerts In New Situations",
    "parameters":{
       "from":{
          "description":"A timestamp from epoch in seconds",
          "type":"Long",
          "required":true
       },
       "aggregation":{
          "default":"none",
          "type":"string",
          "static_mapping":[
             {
```

```
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Accumulate",
                  "value":"accumulate"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getNewEventsPerAlertsStats",
      "description":"returns the number of new events divided by the number
   of new alerts over time",
      "display_name":"Reduction From Events To Alert",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Accumulate",
                  "value":"accumulate"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getNewAlertsPerSituationsStats",
      "description":"returns the number of new alerts divided by the number
   of new situations over time",
      "display_name":"Reduction From Alerts To Situations",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
```

```
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Accumulate",
                  "value":"accumulate"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getNewEventsPerSituationsStats",
      "description":"returns the number of new events divided by the number
   of new situations over time",
      "display_name":"Reduction From Events To Situations",
      "parameters":{
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Accumulate",
                  "value":"accumulate"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getReassignedSituationsPerTeamStats",
      "description":"returns the number of reassigned situations of a team
   over time",
      "display_name":"Reassigned Situations by Team",
      "parameters":{
         "teams":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getTeams",
               "value":"team_id"
```

```
            },
            "type":"mapped",
            "required":false
         },
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Sum",
                  "value":"sum"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getSeveritySituationPerTeamStats",
      "description":"returns the number of active situations with specified
   severity and team over time",
      "display_name":"Open Situations by Severity by Team",
      "parameters":{
         "severity":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getSeverities",
               "value":"severity_id"
            },
            "type":"mapped",
            "required":"false"
         },
         "teams":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getTeams",
               "value":"team_id"
            },
            "type":"mapped",
            "required":false
         },
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
```

```
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Sum",
                  "value":"sum"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getStatusSituationPerTeamStats",
      "description":"returns the number of situations with a specified status
   and team over time",
      "display_name":"Open Situations by Status by Team",
      "parameters":{
         "teams":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getTeams",
               "value":"team_id"
            },
            "type":"mapped",
            "required":false
         },
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Sum",
                  "value":"sum"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "status":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getStatuses",
               "value":"status_id"
```

```
            },
            "type":"mapped",
            "required":"false"
         }
      }
   },
   {
      "endpoint":"getServiceSituationPerTeamStats",
      "description":"returns the number of active situations with specified
   service and team over time",
      "display_name":"Open Situations by Service by Team",
      "parameters":{
         "teams":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getTeams",
               "value":"team_id"
            },
            "type":"mapped",
            "required":true
         },
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Sum",
                  "value":"sum"
               }
            ],
            "required":false
         },
         "services":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getServices",
               "va* Connection #0 to host freida7 left intact
   lue":"service_id"
            },
            "type":"mapped",
            "required":"true"
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getMTTAPerTeamStats",
      "description":"returns the mean time to acknowledge a situation of a
   team over time",
      "display_name":"Mean Time To Acknowledge by Team",
```

```
      "parameters":{
         "teams":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getTeams",
               "value":"team_id"
            },
            "type":"mapped",
            "required":false
         },
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
               },
               {
                  "display_value":"Sum",
                  "value":"sum"
               }
            ],
            "required":false
         },
         "to":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         }
      }
   },
   {
      "endpoint":"getMTTRPerTeamStats",
      "description":"returns the mean time to resolve a situation of a team
   over time",
      "display_name":"Mean Time To Resolve by Team",
      "parameters":{
         "teams":{
            "mapping":{
               "display_value":"name",
               "endpoint":"getTeams",
               "value":"team_id"
            },
            "type":"mapped",
            "required":false
         },
         "from":{
            "description":"A timestamp from epoch in seconds",
            "type":"Long",
            "required":true
         },
         "aggregation":{
            "default":"none",
            "type":"string",
            "static_mapping":[
               {
                  "display_value":"None",
                  "value":"none"
```

```
            },
            {
               "display_value":"Sum",
               "value":"sum"
            }
         ],
         "required":false
      },
      "to":{
         "description":"A timestamp from epoch in seconds",
         "type":"Long",
         "required":true
      }
   }
},
{
   "endpoint":"getReoccurringSituationPerTeamStats",
   "description":"returns the percentage of reoccurring situations of a
team over time",
   "display_name":"Reoccurring situations Per Team",
   "parameters":{
      "teams":{
         "mapping":{
            "display_value":"name",
            "endpoint":"getTeams",
            "value":"team_id"
         },
         "type":"mapped",
         "required":false
      },
      "from":{
         "description":"A timestamp from epoch in seconds",
         "type":"Long",
         "required":true
      },
      "aggregation":{
         "default":"none",
         "type":"string",
         "static_mapping":[
            {
               "display_value":"None",
               "value":"none"
            },
            {
               "display_value":"Sum",
               "value":"sum"
            }
         ],
         "required":false
      },
      "to":{
         "description":"A timestamp from epoch in seconds",
         "type":"Long",
         "required":true
      }
   }
},
{
   "endpoint":"getCommentCountPerTeamStats",
   "description":"returns the number of comments posted on situations by
team members over time",
   "display_name":"Number of Comments by Team",
   "parameters":{
```

```
        "teams":{
           "mapping":{
              "display_value":"name",
              "endpoint":"getTeams",
              "value":"team_id"
           },
           "type":"mapped",
           "required":false
        },
        "from":{
           "description":"A timestamp from epoch in seconds",
           "type":"Long",
           "required":true
        },
        "aggregation":{
           "default":"none",
           "type":"string",
           "static_mapping":[
              {
                 "display_value":"None",
                 "value":"none"
              },
              {
                 "display_value":"Sum",
                 "value":"sum"
              }
           ],
           "required":false
        },
        "to":{
           "description":"A timestamp from epoch in seconds",
           "type":"Long",
           "required":true
         }
      }
   }
]
```

### getStatusSituationStats

A GET request that returns the number of Situations by status.

Request Argument

| Name | Type | Description |
|---|---|---|
| **status** | Array | An array of status ids. This is optional. If not given, it returns the default set of statuses: Opened, Unassigned, Assigned, Acknowledged, Unacknowledged, Resolved. |
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | "sum" - for an aggregation of all teams provided.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the status |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br>Data point: Number of Situations for each status.<br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br>The time range you provide determines number of data points per time unit the request returns. For example:<br>Less than 1 week: Returns the number of Situations each hour in the time period.<br>1 week to 1 month: Returns the number of Situations each day in the time period.<br>1 month to 1 year: Returns the number of Situations each week in the time period.<br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of opened and assigned Situations from 15.27pm on Sunday, January 14th until 15.27pm on Monday, 15th January 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getStatusSituationStats" --data-urlencode
'from=1515943678' --data-urlencode 'to=1516030078' --data-urlencode
'status=[1, 2]' --data-urlencode 'aggregation=sum'
```

Response Example

A successful request returns the number of Situations for each status:

```
[{
        "datapoints": [
                [32.0, 1516008478000],
                [54.0, 1516030078000]
                [68.0, 1516030078000]
                [82.0, 1516030078000]
                [88.0, 1516030078000]
        ],
        "target": "Opened"
}, {
        "datapoints": [
                [5.0, 1515947278000],
                [12.0, 1515958078000],
                [25.0, 1515976078000],
                [31.0, 1515994078000],
                [40.0, 1516015678000]
        ],
        "target": "Assigned"
}]
```

### getSystemSituationStats

A GET request that returns the number of active Situations in the system over time.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |

| | | |
|---|---|---|
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "System" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations for each status.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of active Situations from 11.09am on Sunday, 17th December until 11.09am on Monday, 18th December 2017:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSystemSituationStats" --data-urlencode
'from=1513508950' --data-urlencode 'to=1513595370'
```

Response Example

A successful response returns the number of active Situations every hour during that time range:

```
[{
        "datapoints": [
                [66.0, 1513657700000],
                [98.0, 1513661300000],
                [102.0, 1513664900000],
                [106.0, 1513668500000],
                [92.0, 1513672100000],
                [88.0, 1513675700000],
                [86.0, 1513679300000],
                [74.0, 1513682900000],
                [85.0, 1513672100000],
                [83.0, 1513675700000],
                [79.0, 1513679300000],
                [68.0, 1513686500000]
        ],
        "target": "Open Situations"
}]
```

### getTopServiceSituationStats

A GET request that returns the number of active Situations impacting a top service over a range of time. Top services are the services that have the most situations impacting them.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | "sum" - for an aggregation of all teams provided.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the service |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations for each status.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of Situations impacting top services between 12pm and midnight on Saturday, 15th September 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getServiceSituationStats" --data-urlencode
'from=1537012800' --data-urlencode 'to=1536969600' --data-urlencode
'aggregation=sum'
```

Response Example

A successful response returns the number of Situations each hour for the 12 hour range:

```
[{
        "datapoints": [
                [10.0, 1538133600000],
                [12.0, 1538133600000],
                [8.0, 1538133600000],
```

```
                [5.0, 1538133600000],
                [9.0, 1538133600000],
                [6.0, 1538133600000],
                [10.0, 1538133600000],
                [13.0, 1538133600000],
                [11.0, 1538133600000],
                [7.0, 1538133600000],
                [9.0, 1538133600000],
                [1.0, 1538133600000]
        ],
        "target": "Web Service"
}, {
        "datapoints": [
                [7.0, 1538133600000],
                [3.0, 1538133600000],
                [6.0, 1538133600000],
                [14.0, 1538133600000],
                [9.0, 1538133600000],
                [8.0, 1538133600000],
                [12.0, 1538133600000],
                [11.0, 1538133600000],
                [8.0, 1538133600000],
                [4.0, 1538133600000],
                [6.0, 1538133600000],
                [3.0, 1538133600000]],
        "target": "Cloud Service"
}]
```

## Team Endpoints

The following endpoints return data statistics relating to your Cisco Crosswork Situation Manager teams:

### getCommentCountPerTeamStats

A GET request that returns the total number of comments each hour for a specific team or teams in a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **teams** | Array | An array of team IDs. This is required. If no teams are provided, the endpoint does not return any data. |
| **from** | Number | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | "sum" - for the sum of the number of the comments for all teams in the array.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **target** | String | Name of the team. |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br>Data point: Number of comments.<br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br>The time range you provide determines the number of data points per time unit the request returns. For example:<br>Less than 1 week: Returns the number of comments each hour in the time period.<br>1 week to 1 month: Returns the number of comments each day in the time period.<br>1 month to 1 year: Returns the number of comments each week in the time period.<br>More than 1 year: Returns the number of comments each month in the time period. |

Request Example

A cURL request to retrieve the total number of comments for three teams each hour over a 24 hour time range from 6am on Wednesday 19th September until 6am on Thursday 20th September 2018:

```
curl -G -u graze:graze -k -v
"https://freida7/graze/v1/getCommentCountPerTeamStats" --data-urlencode
'teams=[1,2,3]' --data-urlencode 'from=1537336800' --data-urlencode
'to=1537423200' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of comments per hour for the Cloud DevOps, Database DevOps and Switch DevOps teams:

```
[
        {"datapoints":[
                [14.0,1537357717000]],
                "target":"Cloud DevOps"},
        {"datapoints":[
                [22.0,1537357717000]],
                "target":"Database DevOps"},
        {"datapoints":[
                [10.0,1537357717000]],
                "target":"Switch DevOps"}
]
```

### getMTTAPerTeamStats

A GET request that returns the mean time to acknowledge (MTTA) a Situation per team for a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **teams** | Array | An array containing a single team ID. You can only provide one team ID for this endpoint. This is mandatory. If no teams are provided, the endpoint does not return any data. |
| **from** | Number | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds.<br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

| | | |
|---|---|---|
| **aggregation** | String | "sum" - for an aggregation of all MTTA times.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Mean Time to Acknowledge (MTTA)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Datapoint: MTTA (seconds) for that bucket.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the MTTA each hour in the time period.<br><br>1 week to 1 month: Returns the MTTA each day in the time period.<br><br>1 month to 1 year: Returns the MTTA each week in the time period.<br><br>More than 1 year: Returns the MTTA each month in the time period. |

Request Example

A cURL command request to find out the MTTA for the Cloud DevOps team over a year from 13.14pm on Monday 31st July 2017 until 13.14.pm on Tuesday 31st July 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getMTTAPerTeamStats" --data-urlencode
'from=1501506840' --data-urlencode 'to=1533042840' --data-urlencode
'teams=[1]'  --data-urlencode 'aggregation=none'
```

Response Example

A successful response shows the MTTA for the year was 3.32 minutes:

```
[{
 "datapoints": [
        [213.0, 1532956486000]
 ],
 "target": "Mean Time to Acknowledge (MTTA)"
}]
```

### getMTTRPerTeamStats

A GET request that returns the mean time to resolve (MTTR) a Situation per team for a given time range.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **teams** | Array | An array containing a single team ID. You can only provide one team ID for this endpoint. This is mandatory. If no teams are provided, the endpoint does not return any data. |
| **from** | Number | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |

| | | |
|---|---|---|
| **to** | Number | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **aggregation** | String | "sum" - for an aggregation of all MTTR times.<br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Mean Time to Resolve (MTTR)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br>Data point: MTTR (seconds) for that bucket.<br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br>The time range you provide determines number of data points per time unit the request returns. For example:<br>Less than 1 week: Returns the MTTR each hour in the time period.<br>1 week to 1 month: Returns the MTTR each day in the time period.<br>1 month to 1 year: Returns the MTTR each week in the time period.<br>More than 1 year: Returns the MTTR each month in the time period. |

Request Example

A cURL request for the MTTR of the Cloud DevOps team from 9.26pm on Monday, November 6th until 2.26am on Tuesday, November 7th 2017:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getMTTRPerTeamStats" --data-urlencode
'teams=[1]'  --data-urlencode 'from=1510003600' --data-urlencode
'to=1510021600' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the MTTR each hour from 9.26pm until 2.26am:

```
[{
        "datapoints": [
                [101.6,1510003600000],
                [180.0,1510007200000],
                [210.6667,1510010800000],
                [85.7083,1510014400000],
                [302.5,1510018000000],
                [150.4286,1510021600000]]
        ],
        "target": "Mean Time to Resolve (MTTR)"
}]
```

### getReassignedSituationsPerTeamStats

A GET request that returns the number of reassigned Situations associated with a team or multiple teams over a given time range. A reassigned Situation is a Situation that a user has assigned to another user at least twice.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **teams** | Array | An array of team IDs. This is required. If no teams are provided, the endpoint does not return any data. |
| **from** | Number | Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **aggregation** | String | "sum" - for an aggregation of all Situations.<br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the team: "<team_name>" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br>Data point: Number of reassigned Situations.<br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br>The time range you provide determines number of data points per time unit the request returns. For example:<br>Less than 1 week: Returns the number of reassigned Situations each hour in the time period.<br>1 week to 1 month: Returns the number of reassigned Situations each day in the time period.<br>1 month to 1 year: Returns the number of reassigned Situations each week in the time period.<br>More than 1 year: Returns the number of reassigned Situations each month in the time period. |

Request Example

A cURL request to retrieve the reassigned Situations for the Cloud DevOps and Application Performance Monitoring teams from August 1st until September 1st 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getReassignedSituationsPerTeamStats" --data-
urlencode 'teams=[1,2]' --data-urlencode 'from=1533103200' --data-
urlencode 'to=1535781600'
```

Response Example

A successful response returns the number of reassigned Situations for each week during that month range for both teams:

```
[{
    "datapoints":[
                [4.9702,1533103200000],
```

```
                [4.9881,1533708000000],
                [5.0655,1534312800000],
                [4.9524,1534917600000],
                [4.9917,1535522400000]],
   "target":"Cloud DevOps"
},
{
   "datapoints":[
                [5.006,1533103200000],
                [5.0,1533708000000],
                [5.131,1534312800000],
                [5.0714,1534917600000],
                [4.8417,1535522400000]],
   "target":"Application Performance Monitoring"
}]
```

### getReoccurringSituationPerTeamStats

A GET request that returns the number of reoccurring Situations associated with a team for a given time range.

Request Argument

| Name | Type | Description |
|---|---|---|
| **teams** | Array | An array of team IDs. This is required. If no teams are provided, the endpoint does not return any data. |
| **from** | Number | Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **aggregation** | String | "sum" - for an aggregation of all Situations.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Reoccurring situations" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of reoccurring Situations.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of reoccurring Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of reoccurring Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of reoccurring Situations each week in the time period. |

| | | |
|---|---|---|
| | | More than 1 year: Returns the number of reoccurring Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of reoccuring Situations from 3pm on Saturday, September 1st until 3pm on Saturday, September 8th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getReoccurringSituationPerTeamStats" --data-
urlencode 'teams=[1,2]' --data-urlencode 'from=1535814000' --data-
urlencode 'to=1536418800' --data-urlencode 'aggregation=none'
```

Response Example

A successful response indicates there were four reoccuring Situations at the time the request was sent:

```
[{"datapoints":[[4.0,1538044321144]],"target":"Reoccurring situations"}]
```

### getServiceSituationPerTeamStats

A GET request that returns the number of Situations impacting each service for a team.

Request Argument

| Name | Type | Description |
|---|---|---|
| **teams** | Array | A single team ID in an array. You can only provide one team ID for this endpoint. This is required. If no teams are provided, the endpoint does not return any data. |
| **from** | Number | Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **aggregation** | String | "sum" - for an aggregation of all services.<br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the team. |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br>Data point: Number of Situations impacting services.<br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br>The time range you provide determines number of data points per time unit the request returns. For example:<br>Less than 1 week: Returns the number of Situations each hour in the time period.<br>1 week to 1 month: Returns the number of Situations each day in the time period.<br>1 month to 1 year: Returns the number of Situations each week in the time period.<br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of Situations associated with the Cloud DevOps team that are impacting the Commerce and Compute services between 12pm and 6pm on Friday, August 10th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getServiceSituationPerTeamStats" --data-
urlencode 'from=1533902400' --data-urlencode 'to=1533924000' --data-
urlencode 'teams=[1]' --data-urlencode 'services=[1, 2]' --data-urlencode
'aggregation=none'
```

Response Example

A successful request returns the number of Situations impacting the services each hour during the six hour time range:

```
[{
        "datapoints":[
                [7.0,1533902400000],
                [18.0,1533906000000],
                [18.0,1533909600000],
                [13.0,1533913200000],
                [9.0,1533916800000],
                [12.0,1533920400000]],
        "target":"Commerce"},
{
        "datapoints":[
                [14.0,1533902400000],
                [15.0,1533906000000],
                [6.0,1533909600000],
                [12.0,1533913200000],
                [1.0,1533916800000],
                [11.0,1533920400000]],
        "target":"Compute"
}]
```

### getSeveritySituationPerTeamStats

A GET request that returns the number of Situations by severity per team for a given time range.

Request Argument

| Name | Type | Description |
|---|---|---|
| **teams** | Array | A single team ID in an array. You can only provide one team ID for this endpoint. This is required. If no teams are provided, the endpoint does not return any data. |
| **severity** | Array | An array of severity IDs. This is optional. If not given, it returns default set of severities: Clear, Indeterminate, Warning, Minor, Major, Critical. |
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last data point returned is the current state datapoint. |
| **aggregation** | String | "sum" - for an aggregation of all teams provided.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for |

| | |
|---|---|
| code | details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the status. |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations per severity.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of clear Situations for the Cloud DevOps team between between 12pm onThursday, August 9th and 12pm on Friday, August 10th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSeveritySituationPerTeamStats" --data-
urlencode 'from=1533816000' --data-urlencode 'to=1533902400' --data-
urlencode 'teams=[1]' --data-urlencode 'severity=[0]' --data-urlencode
'aggregation=none'
```

Response Example

A successful response returns the number of clear Situations each hour over the past 24 hours:

```
[{
        "datapoints":[
                [13.0,1533816000000],
                [14.0,1533819600000],
                [6.0,1533823200000],
                [10.0,1533826800000],
                [14.0,1533830400000],
                [5.0,1533834000000],
                [19.0,1533837600000],
                [17.0,1533841200000],
                [4.0,1533844800000],
                [13.0,1533848400000],
                [7.0,1533852000000],
                [15.0,1533855600000],
                [6.0,1533859200000],
                [10.0,1533862800000],
                [16.0,1533866400000],
                [20.0,1533870000000],
                [19.0,1533873600000],
                [15.0,1533877200000],
                [15.0,1533880800000],
                [5.0,1533884400000],
                [20.0,1533888000000],
                [3.0,1533891600000],
                [1.0,1533895200000],
                [4.0,1533898800000]],
```

```
        "target":"Clear"
}]
```

### getStatusSituationPerTeamStats

A GET request that returns the number of Situations by status for a team over a given time range.

Request Argument

| Name | Type | Description |
|---|---|---|
| **teams** | Array | A single team ID in an array. You can only provide one team ID for this endpoint. This is required. If no teams are provided, the endpoint does not return any data. |
| **status** | Array | An array of status IDs. Valid status IDs are 1 (opened), 3 (assigned), 4 (acknowledged), 5 (unacknowledged) and 8 (resolved). Other integers do not return any data.<br><br>If left empty, the request returns the number of Situations for all statuses. |
| **from** | Number | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **aggregation** | String | "sum" - for an aggregation of all teams provided.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the team. |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations for each status.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request with to return all Situations by status for the Cloud DevOps team from 8.30am until 2.30pm on Saturday, September 1st 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getStatusSituationPerTeamStats" --data-
urlencode 'from=1535790600' --data-urlencode 'to=1535812200' --data-
urlencode 'teams=[1]' --data-urlencode 'status=[]' --data-urlencode
'aggregation=none'
```

Response Example

A successful response returns the number of Situations by status each hour for the six hour range:


```
[
        {"datapoints":[
                [19.0,1535790600000],
                [20.0,1535794200000],
                [17.0,1535797800000],
                [18.0,1535801400000],
                [17.0,1535805000000],
                [17.0,1535808600000]],
                "target":"Opened"},
        {"datapoints":[
                [3.0,1535790600000],
                [7.0,1535794200000],
                [4.0,1535797800000],
                [10.0,1535801400000],
                [10.0,1535805000000],
                [2.0,1535808600000]],
                "target":"Assigned"},
        {"datapoints":[
                [3.0,1535790600000],
                [5.0,1535794200000],
                [10.0,1535797800000],
                [3.0,1535801400000],
                [5.0,1535805000000],
                [2.0,1535808600000]],
                "target":"Acknowledged"},
        {"datapoints":[
                [3.0,1535790600000],
                [3.0,1535794200000],
                [4.0,1535797800000],
                [3.0,1535801400000],
                [3.0,1535805000000],
                [2.0,1535808600000]],
                "target":"Unacknowledged"},
        {"datapoints":[
                [46.0,1535790600000],
                [48.0,1535794200000],
                [32.0,1535797800000],
                [48.0,1535801400000],
                [34.0,1535805000000],
                [36.0,1535808600000]],
                "target":"Resolved"}
]
```


### getTeamSituationStats

A GET request that returns the number of active Situations assigned to a team for a given time range.

Request Argument

| Name | Type | Description |
|---|---|---|
| **teams** | Array | An array of team IDs. This is optional. If no teams are provided, the endpoint does not return any data. |
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

| | | |
|---|---|---|
| **aggregation** | String | "sum" - for an aggregation of all teams provided.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the team . |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations for each status.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to return the number of active Situations assigned to the Cloud DevOps and Application Performance Monitoring teams from midnight until 6am on Monday, 20th August 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamSituationStats" --data-urlencode
'teams=[1,2]' --data-urlencode 'from=1534723200' --data-urlencode
'to=1534744800' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of Situations assigned each hour to each team for the six hour range:

```
[
        {"datapoints":[
                [30.0,1534723200000],
                [20.0,1534726800000],
                [24.0,1534730400000],
                [19.0,1534734000000],
                [28.0,1534737600000],
                [23.0,1534741200000]],
                "target":"Cloud DevOps"},
        {"datapoints":[
                [26.0,1534723200000],
                [29.0,1534726800000],
                [15.0,1534730400000],
                [29.0,1534734000000],
                [25.0,1534737600000],
                [22.0,1534741200000]],
                "target":"Application Performance Monitoring"}]
```

### getTopTeamSituationStats

A GET request that returns the number of active Situations assign to top teams over a given range of time. Top teams are those teams with the highest number of assigned Situations.

Request Argument

| Name | Type | Description |
|---|---|---|
| **from** | Number | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | "sum" - for an aggregation of all teams provided.<br><br>"none" - for no aggregation of results. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | The name of the team. |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations for each status.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations each week in the time period.<br><br>More than 1 year: Returns the number of Situations each month in the time period. |

Request Example

A cURL request to retrieve the number of Situations impacting top teams between 6am and 12pm onWednesday, 1st August 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamSituationStats"  --data-urlencode
'from=1533103200' --data-urlencode 'to=1533124800' --data-urlencode
'aggregation=sum'
```

Response Example

A successful response returns the number of Situations per hour for the six hour time time range:

```
[{
        "datapoints": [
                [2.0, 1538133780000],
                [9.0, 1538133780000],
                [5.0, 1538133780000],
                [4.0, 1538133780000],
```

```
                [3.0, 1538133780000],
                [1.0, 1538133780000]
        ],
        "target": "Cloud DevOps"
}, {
        "datapoints": [
                [8.0, 1538133780000],
                [2.0, 1538133780000],
                [6.0, 1538133780000],
                [7.0, 1538133780000],
                [5.0, 1538133780000],
                [3.0, 1538133780000]
        ],
        "target": "Application Performance Monitoring"
}]
```

## User Endpoints

The following endpoints return data statistics relating to your Cisco Crosswork Situation Manager users:

### getAlertsMarkedPRCPerUserStats

A GET request that returns the total number of alerts marked with probable root cause (PRC) feedback by each user.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Alerts Marked PRC (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of alerts marked with PRC feedback by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of alerts marked with PRC feedback each hour in the time period. |

| | | |
|---|---|---|
| | | 1 week to 1 month: Returns the number of alerts marked with PRC feedback each day in the time period.<br><br>1 month to 1 year: Returns the number of alerts marked with PRC feedback each week in the time period.<br><br>More than 1 year: Returns the number of alerts marked with PRC feedback each month in the time period. |

Request Example

A cURL request to return the number of alerts marked with PRC feedback to users 5 and 6 from 8am until 2pm on on Friday, 28th September 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getAlertsMarkedPRCPerUserStats" --data-
urlencode 'users=[5, 6]' --data-urlencode 'from=1538121620' --data-
urlencode 'to=1538143220' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of alerts that users Alice and Ian have marked with PRC feedback each hour during the time range:

```
[{
        "datapoints":[
                [22.0,1538121620000],
                [18.0,1538125220000],
                [30.0,1538128820000],
                [23.0,1538132420000],
                [29.0,1538136020000],
                [28.0,1538139620000]]
        ],
        "target":"Alerts Marked PRC (Alice Anderson)"
}
{
        "datapoints":[
                [34.0,1538121620000],
                [20.0,1538125220000],
                [35.0,1538128820000],
                [21.0,1538132420000],
                [19.0,1538136020000],
                [10.0,1538139620000]]
        ],
        "target":"Alerts Marked PRC (Ian Ince)"
}]
```

### getAcknowledgedSituationsPerUserStats

A GET request that returns the number of Situations acknowledged by a specific user or users within a given time range.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

| | | | |
|---|---|---|---|
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Acknowledged Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations acknowledged by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations acknowledged each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations acknowledged each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations acknowledged each week in the time period.<br><br>More than 1 year: Returns the number of Situations acknowledged each month in the time period. |

Request Example

A cURL request to return the number of Situations acknowledged by user Bob from 9am on Friday 28th September until 3pm on Friday 28th September 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getAcknowledgedSituationsPerUserStats" --data-
urlencode 'users=[6]' --data-urlencode 'from=1538121620' --data-urlencode
'to=1538143220'
```

Response Example

A successful response returns the number of Situations acknowledged by Bob each hour during that time frame:

```
[{
        "datapoints":[
                [2.0,1538121620000],
                [3.0,1538125220000],
                [0.0,1538128820000],
                [2.0,1538132420000],
                [2.0,1538136020000],
                [2.0,1538139620000]
        ],
        "target":"Acknowledged Situations (Bob Bowden)"
}]
```

### getAssignedSituationsPerUserStats

A GET request that returns the number of Situations assigned to a specific user or users within a given time range.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Assigned Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations assigned to the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations assigned each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations assigned each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations assigned each week in the time period.<br><br>More than 1 year: Returns the number of Situations assigned each month in the time period. |

Request Example

A cURL request to return the number of Situations assigned to users 10 and 11 from 11pm on Tuesday, 25th September 2018 until 11pm on Wednesday, 26th September 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getAssignedSituationsPerUserStats" --data-
urlencode 'users=[10,11]' --data-urlencode 'from=1537916400' --data-
urlencode 'to=1538002799' --data-urlencode 'aggregation=sum'
```

Response Example

A successful response returns the number of Situations assigned to the users Frank and Dave each hour during the time range:


```
[{
        "datapoints":[
                [10.0,1537916400000],
                [5.0,1537920000000],
                [7.0,1537923600000],
                [7.0,1537927200000],
                [7.0,1537930800000],
                [1.0,1537934400000],
                [5.0,1537938000000],
                [6.0,1537941600000],
                [9.0,1537945200000],
                [9.0,1537948800000],
                [7.0,1537952400000],
                [8.0,1537956000000]
        ],
        "target":"Assigned Situations (Frank Fuller/Dave Danton)"
}]
```


### getChatOpsToolExecutedPerUserStats

A GET request that returns the number of ChatOps tools executed by a user each hour within a given time range.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Chat Ops Tools executed (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of ChatOps tools executed by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example: |

| | | |
|---|---|---|
| | | Less than 1 week: Returns the number of ChatOps tools executed each hour in the time period.<br><br>1 week to 1 month: Returns the number of ChatOps tools executed each day in the time period.<br><br>1 month to 1 year: Returns the number of ChatOps tools executed each week in the time period.<br><br>More than 1 year: Returns the number of ChatOps tools executed each month in the time period. |

Request Example

A cURL request to retrieve the total number of ChatOps tools executed by user 5 from 11pm on Sunday, 14th October until 11pm on Monday, 15th October 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getChatOpsToolExecutedPerUserStats" --data-
urlencode 'users=[5]' --data-urlencode 'from=1539558000' --data-urlencode
'to=1539644399' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of ChatOps tools executed by the user Max each hour:

```
[{
        "datapoints":[
                [6.0,1539558000000],
                [24.0,1539561600000],
                [1.0,1539565200000],
                [0.0,1539568800000],
                [14.0,1539572400000],
                [10.0,1539576000000],
                [4.0,1539579600000],
                [12.0,1539583200000],
                [25.0,1539586800000],
                [8.0,1539590400000],
                [0.0,1539598043846]
        ],
                "target":"ChatOps Tools executed (Max Matthews)"
}]
```

### getClosedSituationsPerUserStats

A GET request that returns the number of Situations that a user has closed each hour within a given time range.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Closed Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations closed by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations closed each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations closed each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations closed each week in the time period.<br><br>More than 1 year: Returns the number of Situations closed each month in the time period. |

Request Example

A cURL request to return the number of Situations closed by user 12 from 6am until midnight on October 1st 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getClosedSituationsPerUserStats" --data-
urlencode 'users=[5]' --data-urlencode 'from=1538373600' --data-urlencode
'to=1538395200' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of Situations closed by user Chris each hour during the time range:

```
[{
        "datapoints":[
                [1.0,1539558000000],
                [1.0,1539561600000],
                [2.0,1539565200000],
                [5.0,1539568800000],
                [0.0,1539572400000],
                [7.0,1539576000000],
                [1.0,1539579600000],
                [0.0,1539583200000],
                [8.0,1539586800000],
                [6.0,1539590400000],
                [0.0,1539594000000],
                [0.0,1539597600000]
        ],
        "target":"Closed Situations (Chris Collins)"
}]
```

### getCommentCountPerUserStats

A GET request that returns the number of comments left by a user or users within a given time range.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Number of Comments (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of comments.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of comments each hour in the time period.<br><br>1 week to 1 month: Returns the number of comments each day in the time period.<br><br>1 month to 1 year: Returns the number of comments each week in the time period.<br><br>More than 1 year: Returns the number of comments each month in the time period. |

Request Example

A cURL request to retrieve the total number of comments made by user 9 and 11 each hour from 11pm on Sunday, 14th October until 11pm on Monday, 15th October 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getCommentsPerUserStats" --data-urlencode
'users=[9,11]' --data-urlencode 'from=1539558000' --data-urlencode
'to=1539644399' --data-urlencode 'aggregation=sum'
```

Response Example

A successful response returns the number of comments made by the users Ian and Sharon each hour:

```
[{
        "datapoints":[
                [6.0,1539558000000],
                [24.0,1539561600000],
                [1.0,1539565200000],
                [0.0,1539568800000],
```

```
                [14.0,1539572400000],
                [10.0,1539576000000],
                [4.0,1539579600000],
                [12.0,1539583200000],
                [25.0,1539586800000],
                [8.0,1539590400000],
                [0.0,1539598043846]
        ],
                "target":"Number of Comments (Ian Ince/Sharon Scott)"
}]
```


### getInvitationsReceivedPerUserStats

A GET request that returns the number of Situation invitations received for a given user each hour within a given time range.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Invitations Received (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Invitations received by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situation invitations each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situation invitations each day in the time period.<br><br>1 month to 1 year: Returns the number of Situation invitations each week in the time period. |

| | | |
|---|---|---|
| | | More than 1 year: Returns the number of Situation invitations each month in the time period. |

Request Example

A cURL request for the number of Situation invitations for users 7 and 8 from midnight on Sunday, 14th October until 6am on Monday, 15th October 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getInvitationsReceivedPerUserStats" --data-
urlencode 'users=[7,8]' --data-urlencode 'from=1539558000' --data-
urlencode 'to=1539583200' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of invitations for users 7 and 8:

```
[{
        "datapoints": [
                [1.0,1539558000000],
                [1.0,1539561600000],
                [2.0,1539565200000],
                [5.0,1539568800000],
                [0.0,1539572400000],
                [7.0,1539576000000],
                [1.0,1539579600000],
                [0.0,1539583200000],
                [8.0,1539586800000],
                [1.0,1539579600000],
                [2.0,1539583200000],
                [0.0,1539586800000],
        ],
        "target": "Invitations Received (Peter Parker/Kat Knight)"
}]
```

### getMTTAPerUserStats

A GET request that returns the mean time it takes a user to acknowledge a Situation within a given time range.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Mean Time to Acknowledge (MTTA) (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br>Data point: MTTA (seconds) for that bucket.<br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br>The time range you provide determines the number of data points per time unit the request returns. For example:<br>Less than 1 week: Returns the MTTA each hour in the time period.<br>1 week to 1 month: Returns the MTTA each day for in time period.<br>1 month to 1 year: Returns the MTTA each week in the time period.<br>More than 1 year: Returns the MTTA each month in the time period. |

Request Example

A cURL request for the MTTA for user 5 from 6.34am until 2.35pm on Tuesday, 25th September 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getMTTAPerUserStats" --data-urlencode
'users=[5]' --data-urlencode 'from=1537857295' --data-urlencode
'to=1537886111' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the MTTA each hour for the user Robert:

```
[{
        "datapoints": [
                [221,1537857295000],
                [960,1537860895000],
                [901,1537864495000],
                [1196,1537868095000],
                [671,1537871695000],
                [1241,1537875295000],
                [556,1537878895000]
        ],
        "target": "Mean Time to Acknowledge (MTTA)(Robert Richards)"
}]
```

### getMTTRPerUserStats

A GET request that returns the mean time it takes a user to resolve a Situation within a given time range.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range you want to collect data from. This is a Unix epoch timestamp in seconds.<br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |

| | | | |
|---|---|---|---|
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Mean Time to Resolve (MTTR) (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: MTTR (seconds) for that bucket.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the MTTR each hour in the time period.<br><br>1 week to 1 month: Returns the MTTR each day in the time period.<br><br>1 month to 1 year: Returns the MTTR each week in the time period.<br><br>More than 1 year: Returns the MTTR each month in the time period. |

Request Example

A cURL request for the MTTR for user 5 from 11pm on Monday, 1st October until 5am on Tuesday, 2nd October 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getMTTRPerUserStats" --data-urlencode
'user=[5]' --data-urlencode 'from=1538434800' --data-urlencode
'to=1538456400' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the MTTR each hour:

```
[{
        "datapoints": [
                [12997.0,1538434800000],
                [14025.0,1538438400000],
                [2969.0,1538442000000],
                [13125.0,1538445600000],
                [11412.0,1538449200000],
                [8264.0,1538452800000]
        ],
        "target": "Mean Time to Resolve (MTTR)(Oscar O'Neill)"
}]
```

### getOpenSituationsPerUserStats

A GET request that returns the number of open Situations assigned to a user at each data point.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|

| | | | |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Open Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of open Situations assigned to the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations assigned each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations assigned each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations assigned each week in the time period.<br><br>More than 1 year: Returns the number of Situations assigned each month in the time period. |

Request Example

A cURL request to return the number of open Situations assigned to user 6 from 9.19am on Monday, 17th September until 16.19am on Monday, 17th September 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getOpenSituationsPerUserStats" --data-
urlencode 'users=[6,7]' --data-urlencode 'from=1537175946' --data-
urlencode 'to=1537201140' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of open Situations assigned to the users Oscar and Olivia each hour during the time range:

```
[{
        "datapoints":[
                [12.0,1537175946000],
```

```
                [8.875,1537262346000],
                [10.0,1537348746000],
                [8.9,1537435146000],
                [10.75,1537521546000],
                [9.25,1537607946000],
                [8.1667,1537694346000]
        ],
        "target":"Open Situations (Oscar O'Neill)"
},
{
        "datapoints":[
                [4.0,1537175946000],
                [5.0,1537262346000],
                [12.0,1537348746000],
                [7.0,1537435146000],
                [3.0,1537521546000],
                [9.0,1537607946000],
                [8.0,1537694346000]
        ],
        "target":"Open Situations (Andrew Anderson)"
}
]
```

### getRatedSituationsPerUserStats

A GET request that returns the number of Situations rated by a user within a given time range.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Rated Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations rated by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example: |

| | | |
|---|---|---|
| | | Less than 1 week: Returns the number of Situations rated each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations rated each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations rated each week in the time period.<br><br>More than 1 year: Returns the number of Situations rated each month in the time period. |

Request Example

A cURL request to return the number of Situations rated by users 5 and 7 from 3:57am until 9:57am on Thursday, October 5th 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getRatedSituationsPerUserStats" --data-
urlencode 'users=[5,7]' --data-urlencode 'from=1538621843' --data-
urlencode 'to=1538643443'
```

Response Example

A successful response returns the number of Situations rated by the users Steve and Charlie each hour during the time range:

```
[{
        "datapoints":[
                [6.0,1538621843000],
                [1.0,1538625443000],
                [6.0,1538629043000],
                [5.0,1538632643000],
                [2.0,1538636243000],
                [5.0,1538639843000]
        ],
        "target":"Rated Situations (Steve Smith)"
  },
  {
        "datapoints":[
                [0.0,1538621843000],
                [3.0,1538625443000],
                [1.0,1538629043000],
                [6.0,1538632643000],
                [6.0,1538636243000],
                [8.0,1538639843000]
        ],
        "target":"Rated Situations (Charlie Copper)"
}]
```

### getReassignedSituationsPerUserStats

A GET request that returns the number of Situations reassigned by a user within a given time range.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together. |

| | | | |
|---|---|---|---|
| | | | "none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Reassigned Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations reassigned by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations reassigned each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations reassigned each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations reassigned each week in the time period.<br><br>More than 1 year: Returns the number of Situations reassigned each month in the time period. |

Request Example

A cURL request to return the number of Situations reassigned by user 5 from 11pm on Sunday, 14th October until 5am on Monday, 15th October 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getReassignedSituationsPerUserStats" --data-
urlencode 'users=[5]' --data-urlencode 'from=1539558000' --data-urlencode
'to=1539579600' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of Situations reassigned by the user Dave each hour during the time range:

```
[{
        "datapoints":[
                [2.0,1539558000000],
                [3.0,1539561600000],
                [0.0,1539565200000],
                [1.0,1539568800000],
                [0.0,1539572400000],
                [3.0,1539576000000],
        ],
        "target":"Reassigned Situations (Dave Danton)"
}]
```

### getResolvedSituationsPerUserStats

A GET request that returns the number of Situations resolved by a user within a given time range.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Resolved Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations resolved by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations resolved each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations resolved each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations resolved each week in the time period.<br><br>More than 1 year: Returns the number of Situations resolved each month in the time period. |

Request Example

A cURL request to return the number of Situations resolved by user 5 from 8.47am until 15.04pm on October 1st 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getResolvedSituationsPerUserStats" --data-
urlencode 'users=[5]' --data-urlencode 'from=1538380070' --data-urlencode
'to=1538402670' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of Situations resolved by the user Alice each hour during the time range:

```
[{
        "datapoints":[
                [5.0,1538380070000],
                [3.0,1538383670000],
                [8.0,1538387270000],
                [0.0,1538390870000],
                [0.0,1538394470000],
                [8.0,1538398070000],
        ],
        "target":"Resolved Situations (Alice Anderson)"
}]
```

### getViewedSituationsPerUserStats

A GET request that returns the number of Situations a user has viewed within a given time range. Cisco Crosswork Situation Manager considers a user to have viewed a Situation if they opened the Situation Room.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Viewed Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations viewed by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of Situations viewed each hour in the time period.<br><br>1 week to 1 month: Returns the number of Situations viewed each day in the time period.<br><br>1 month to 1 year: Returns the number of Situations viewed each week in the time period.<br><br>More than 1 year: Returns the number of Situations viewed each month in the time |

| | | |
|---|---|---|
| | | period. |

Request Example

A cURL request to return the number of viewed Situations by user 7 from 9am until 3pm on Thursday, 20th September 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getViewedSituationsPerUserStats" --data-
urlencode 'users=[7]' --data-urlencode 'from=1537434000' --data-urlencode
'to=1537455600' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of Situations viewed by the user Charlie each hour during the time range:

```
[{
        "datapoints":[
                [16.0,1537434000000],
                [26.0,1537437600000],
                [18.0,1537441200000],
                [34.0,1537444800000],
                [18.0,1537448400000],
                [11.0,1537452000000]
        ],
        "target":"Viewed Situations (Charlie Cooper)"
}]
```

### getWorkedSituationsPerUserStats

A GET request that returns the number of Situations a user has worked on within a given time range.

Cisco Crosswork Situation Manager considers a user to have worked on a Situation if any of the following apply:

- User has been assigned a Situation.
- User has been invited to a Situation.
- User has a left a comment on a Situation.
- User has closed a Situation.
- User has resolved a Situation.
- User has executed a ChatOps tool on a Situation.
- User has rated a Situation.
- User has added PRC data to alerts in a Situation.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **users** | Array | Yes | An array of user IDs. If no users are provided, the endpoint does not return any data. |
| **from** | Number | Yes | Start of the reporting time range. This is a Unix epoch timestamp in seconds. |
| **to** | Number | Yes | End of the reporting time range. This is a Unix epoch timestamp in seconds.<br><br>If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint. |
| **aggregation** | String | No | "sum" - adds data points together.<br><br>"none" - no aggregation of data points. Defaults to "none" if you provide no |

| | | | |
|---|---|---|---|
| | | | aggregation option. |

Response

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure. See HTTP Status and Error Codes for details. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| **target** | String | "Worked Situations (full name)" |
| **datapoints** | Number array | An array of data points. Each data point is an array in the format [data point, timestamp]:<br><br>Data point: Number of Situations worked on by the user.<br><br>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).<br><br>The time range you provide determines the number of data points per time unit the request returns. For example:<br><br>Less than 1 week: Returns the number of worked Situations each hour in the time period.<br><br>1 week to 1 month: Returns the number of worked Situations each day in the time period.<br><br>1 month to 1 year: Returns the number of worked Situations each week in the time period.<br><br>More than 1 year: Returns the number of worked Situations each month in the time period. |

Request Example

A cURL request to return the number of Situations worked on by user 5 from 12:22pm on Thursday 30th August until 8:22am Friday 31st August 2018:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getWorkedSituationsPerUserStats" --data-
urlencode 'users=[5]' --data-urlencode 'from=1535628143' --data-urlencode
'to=1535700143' --data-urlencode 'aggregation=none'
```

Response Example

A successful response returns the number of Situations worked by the user Chris each hour during the time range:

```
[{
        "datapoints":[
                [12.0,1535628143000],
                [25.0,1535631743000],
                [33.0,1535635343000],
                [14.0,1535638943000],
                [1.0,1535642543000],
                [4.0,1535646143000],
                [9.0,1535649743000],
                [6.0,1535653343000],
                [37.0,1535656943000],
                [31.0,1535660543000],
                [19.0,1535664143000],
                [35.0,1535667743000],
                [36.0,1535671343000],
                [28.0,1535674943000],
                [30.0,1535678543000],
                [19.0,1535682143000],
```

```
                [21.0,1535685743000],
                [30.0,1535689343000],
                [35.0,1535692943000],
                [30.0,1535696543000]
            ],
            "target":"Worked Situations (Chris Cole)"
    }]
```

### HTTP Status and Error Codes

The Graze API returns the following HTTP status and error codes for successful and unsuccessful requests:

| HTTP Code | Meaning |
|---|---|
| 200 | Successful request. |
| 400 | Incorrectly formatted request. |
| 401 | A request with an invalid or expired **auth_code**. |
| 403 | Forbidden request. |
| 404 | Not found, for example, the **sitn_id** could not be found because it does not exist. |
| 500 | Failed request, for example, due to an invalid **sitn_id**. |

## Moobot Modules

Within Cisco Crosswork Situation Manager data processing, Moogfarmd Moolets, LAMs and integrations use simple computer programs called "bots" to perform automated tasks. A Moobot is a JavaScript file that is loaded at startup by a Moolet. The Moobot exposes logic and data flow, which you can control in JavaScript, relevant to the necessary function. LAMbots perform a similar function for LAMs and integrations.Moolets

Moobots expose the function of the Moolets allowing for extensive customization, for example in the Alert Rules Engine where the Moobot is used to perform automation.

### Threads and Global Scope

Cisco Crosswork Situation Manager is built to handle high scale environments, so individual JavaScript MooBots are run in a multi-threaded fashion. For example, if a Moolet has ten threads, there will be ten instances of the MooBot running. This supports high throughput of Events through the Moobot, particularly, when they are doing complex processing. However, it does have important implications for the JavaScript concerning where the global scope (or context) for the JavaScript program for the MooBot resides. In principle, each Moobot has its own independent global scope. It is impossible for one Moobot's logic to interact and affect another instance of the Moobot logic. To allow necessary communication between individual Moobot instances there are utility modules such as the Constants module.

### Moobot Modules

You can use the available Moobot modules to perform these functions:

| Module | Description |
|---|---|
| Config | Read configuration files within LAMbots and Moobots. |
| Constants | Build a key value dictionary shared across Moobots. |
| Events | Set the types of Event that interest a Moobot. |
| ExternalDb | Access external relational databases. |
| LoggerConfigure Logging | Write log messages to the common Moogfarmd log file. See Configure Logging. |

| | |
|---|---|
| Mailer | Send an email in response to events occurring in Cisco Crosswork Situation Manager. |
| MoogDb V2 | Query and manipulate a variety of entities in the Cisco Crosswork Situation Manager database, including alerts and Situations. |
| Moolet Informs | Can send update messages from one Moolet to other Moolets. |
| Process | Run and control the execution of other processes. |
| RabbitMQ | Allows you to broadcast information on a RabbitMQ bus. |
| REST.V2 | Access an external RESTful API via HTTP to post, read, or delete data. |
| Utilities | Escape and unescape XML strings, convert an XML string to a JSON object and vice versa. |

To use these modules, at the top of the Moobot js file, define their variables using the **`loadModule`** method.

You can also load load external JavaScript modules using the **`loadModule`** method. See below.

### Examples

Throughout this section, all examples will use **`AlertBuilder.js`** to explain how Moobots function.

Step 1

When the Alert Builder starts and creates an instance of the Moolet, it creates a Moobot for every threaded instance of the Moolet. The first action undertaken by a Moobot is to load a system wide default file called **`MooB.js`**. This file pushes into the Global Scope using a closure, some shared functionality, which you can take advantage of in the Moobot. You should never edit **`MooB.js`** as the file is linked to the internal implementation of the Moobots.

Step 2

The preload statements in the **`MooB.js`** closure instruct a Moobot to load into its Global Scope the available modules. For example, they can be used to:

- Change and create structure in the moogdb database
- Listen for specific Events in the system
- Push Events out
- Log to the common log file output
- Communicate using communication methodologies such as tweets, email etc.

Before you can use any of the built in modules that correspond to the functionality Cisco provides, you need the **`preload()`** method in the global object (MooB.js) to load the required modules.

The object exposes an API that you can use to add functionality into the system. In the example above, "**`Process`**" has a number of functions that you can call which allow the Moolet to run processes in the system.

After loading and running the **`MooB.js`** closure in the Moobot, the full Moobot user definable JavaScript file is loaded and run. It is important to understand from a JavaScript concept that it is executed at start-up. The reason for executing the script at start-up is to load any Event driven callbacks, and initialization code inside of the Moobot. For example in the Alert Builder, for a new Event arriving in the Moolet, Cisco Crosswork Situation Manager needs to know which functionality inside of the Moobot to run.

## Using External Modules

Moobots can load external JavaScript modules. This means that modules can be reused as generic functions in multiple Moobots.

To do this:

- Add the external JavaScript module file (**`BotExampleModule.js`**) in the **`$MOOGSOFT_HOME/bots/moobots`** or the **`$MOOGSOFT_HOME/contrib`** directory

- Load the external JavaScript module in the Moobot by adding a line at the beginning (relative paths are supported), for example:

  **MooBot.loadModule('BotExampleModule.js');**

The example below shows the external JavaScript module (**BotExampleModule.js**). It defines a class which takes an Alert and prints out a message:

```
function CPrinter()
{
    var mLogger=MooBot.loadModule('Logger');
    var self=
    {
        prettyPrint: function(alert)
        {
            mLogger.info("This is a print of " + alert.value("alert_id") + "
other info");
        }
    };
    var F=function() {};
    F.prototype=self;
    return( new F() );
}
```

The **AlertMgr.js** Moobot loads the external JavaScript module **BotExampleModule.js** and uses the function **CPrinter** (from the external JavaScript module) to send Alert details to a remote service:

```
MooBot.loadModule('BotExampleModule.js');
var printer = new CPrinter();

function newAlert(alert)
{
    printer.prettyPrint(alert);
}
```

## onLoad Function

Moobots can include an onLoad function to allow commands to be run once on startup per Moobot instance. This can be used to initialize internal variables, such as **dbTypes**, as shown in the code example below:

```
var dbTypes = null;
function onLoad()

{
    dbTypes = {
        employees: {
            type: 'mySql',
            host: '192.168.1.141',
            port: '3306',
            database: 'emp_db'
        },
        customers: {
            type: 'sqlServer',
            host: '213.32.112.17',
            database: 'customers',
            user: 'sa',
            encrypted_password:
'0rJGl5oCWpmE9Hbk32sxFgxlQV3O5cx2bx1vKNOM7YA='
        }
```

```
        };
    }
```

## Config

- Before You Begin
- Best Practice
- Error Reporting
- Examples

The Config bot module allows you to read configuration files within LAMbots and Moobots.

It retrieves valid JSON configuration files found in **$MOOGSOFT_HOME/config** and performs a direct read from the file system before delivering the JSON Object to the calling bot. The module is available for all bots but can only be used for reading and storing global configuration files.

### Before You Begin

Before you use the Config bot module, ensure the following conditions are met:

- The configuration file is in valid JSON
- The configuration file is in **$MOOGSOFT_HOME/config**
- The configuration is present on the file system as the process running the bot

### Best Practice

Follow these guidelines when using the Config bot module:

- Use the module within the constraints of the **OnLoad** function.
- Note that making multiple calls to the module may impact the performance of the bot.
- Keep custom configuration files in a subdirectory of **$MOOGSOFT_HOME/config** and name them appropriately.
- Comment custom configuration files extensively so other users can understand the context of their use.

### Error Reporting

The following error messages are returned if the configuration file cannot be opened, the contents returned are null or if the JSON is invalid:

```
INFO :[CJSONCodec.java]:813 +|java.io.FileNotFoundException:
/export/src/incident/build/config/bad.conf (No such file or directory):
Unable to open file /export/src/incident/build/config/bad.conf|+

WARN :[CJSONCodec.java]:105 +|Failed to parse file
/export/src/incident/build/config/bad.conf, returned null contents|+

WARN :[CConfigModule.java]:112 +|File
[/export/src/incident/build/config/bad.conf] is either missing, unreadable
or is not valid JSON.|+
```

### Examples

If you want to create a URL that links to Cisco Crosswork Situation Manager Situations, you can use the Config bot module to dynamically retrieve the base URL of the Cisco Crosswork Situation Manager instance from **servlets.conf**. For example:

```
var config = MooBot.loadModule('Config');
....
var servletsConf = config.getConfig('servlets.conf');
if (servletsConf) {
```

```
        moogURL = servletsConf.webhost;
            }
```

## Constants

- Reference Guide
    - put
    - get
    - contains
    - remove
    - eventType
- Examples

Each Moobot runs in its own thread and instances of Moobots are independent of each other. The Constants module enables you to share logic, states or flags between Moobots. You can build a key value dictionary mapping that is shared across Moobot instances.

There are many system wide defined Constants that are used in the Events module to define which event to listen for. See the event types table below for more information.

The Constants module is available to load into any standard Moobot.

To use, define a new global object **`constants`** at the top of a Moobot JavaScript file:

```
var constants = MooBot.loadModule('Constants');
```

## Reference Guide

You can use the following methods in the RabbitMQ Moobot module.

### put

Associated the specified value with the specified key. Replaces the mapping for an existing key.

Request Argument

| Name | Type | Description |
|---|---|---|
| **`key`** | String | The key to associate with the value. |
| **`value`** | Object | The value to associate with the key. |

Return Parameter

None.

### get

Returns the value mapped to a specified key.

Request Argument

| Name | Type | Description |
|---|---|---|
| **`key`** | String | The key for which to retrieve the value. |

Return Parameter

| Type | Description |
|---|---|
| **`Object`** | The value to which the key is mapped, or null if no mapping exists. |

**contains**

Returns a positive response if the module contains an object with the specified name.

Request Argument

| Name | Type | Description |
|---|---|---|
| **name** | String | The Object name. |

Return Parameter

| Type | Description |
|---|---|
| **Boolean** | True if the module contains the named object, otherwise false. |

**remove**

Removes the value mapped to the specified key.

Request Argument

| Name | Type | Description |
|---|---|---|
| **key** | String | The key for which the value is to be removed. |

Return Parameter

None.

**eventType**

Returns the value of a specified event type.

Request Argument

| Name | Type | Description |
|---|---|---|
| **name** | String | The name of the event type. See the list below. |

**Event Types**

| Name | Passed Value | Description |
|---|---|---|
| **E_LamEvent** | **"Event"/"Events"** | Raw event from a LAM |
| **E_NewAlert** | **"Alert"/"Alerts"** | New alert |
| **E_AlertUpdate** | **"AlertUpdate"** | Alert update |
| **E_CloseAlert** | **"AlertClose"** | Close alert |
| **E_NewComment** | **"Comment"** | New comment |
| **E_NewFeedback** | **"Feedback"** | New feedback |
| **E_NewSig** | **"Sig"** | New Situation |
| **E_SigClose** | **"SigClose"** | Close Situation |
| **E_SigUpdate** | **"SigUpdate"** | Updated Situation |
| **E_SigStatus** | **"SigStatus"** | Situation status |
| **E_SigAction** | **"SigAction"** | Situation action |
| **E_ThreadEntry** | **"ThreadEntry"** | A thread entry |
| **E_NewThreadEntry** | **"NewThreadEntry"** | A new thread entry |
| **E_Summary** | **"Summary"** | System summary |

| | | |
|---|---|---|
| **E_Invite** | **"Invitation"** | Situation Room invitation |
| **E_User** | **"User"** | Username |
| **E_Unknown** | **"Unknown"** | An uncategorised event (error condition) |

Return Parameter

| Type | Description |
|---|---|
| CEvent | An object containing the value of the specified event type. |

### Examples

The following example in AlertBuilder.js shows the Constants module with two methods that allow you to post and retrieve values from a shared scratchpad.

```
var count=0
constants.put("counter",count);
```

The variable count is set to 0 and stored using the label **counter**.

- **put ()** takes the name of the variable you want to store **counter**, and the variable **count**.

You can later retrieve a value by calling the method **get ()** and passing the name of the shared attribute you want, which is returned as a JavaScript local variable.

```
var count_val=constants.get("counter");
count_val++;
constants.put("counter",count_val);
```

- **get** takes the name of the shared attribute **counter**.
- **count_val** is incremented.
- **put** takes the name of the variable to store, **counter**, and the incremented value **count_val**.

  If nothing is stored in **counter**, the Moobot returns **null**.

The following example passes the name of an event and returns a system wide constant that identifies that type of event when using the Events module.

```
constants.eventType("Event")
```

### Events

The Events Moobot module allows you to make a Moobot driven by the occurrence of events by defining the type of event that interests the Moobot.

The Events module is available to load into any standard Moobot.

To use, at the top of a Moobot js file, define a new global object **events** to load the Events module:

```
var events = MooBot.loadModule('Events');
```

### Method

events.onEvent

The Events module has only one method, **onEvent**. This method points the Moobot to a supplied JavaScript function, which is called when a specified event type occurs.

The parameters to the called function depend on the type of event that you are listening for.

In a Moobot, this method is typically the last line in a script.

The type of event adaptor chosen is specific to the type of Moobot you are building.

events.onEvent()

Takes the name of a valid JavaScript function in a Moobot and also event code (from the Constants module **`eventType`**), and returns an event adaptor object

Request Arguments

| Name | Type | Description |
|---|---|---|
| **`functionName`** | String | The name of a valid JavaScript function in the Moobot that is called when the event arrives. |
| **`type`** | CEvent | **`eventType`** event code that specifies what type of event the Moobot is listening for. It is typically from the Constants module. |

Return Parameter

| Name | Type | Description |
|---|---|---|
| **`CEventAdaptor`** | Object | An event adaptor object. Made active with the **`listen ()`** function in-line to listen for the event type. |

Example

For the AlertBuilder MooBot:

```
events.onEvent("newEvent",constants.eventType("Event")).listen();
```

- Call the newEventJavaScript function.
- Define the Event type Event (from the Constants module), which responds to events put on the Message Bus by a LAM.
- Call the listenfunction in-line to listen for the event type.

When the Moolet starts and loads this events Moobot, it's JavaScript file executes, initializing the Moobot to respond in an event-driven way to events arriving.

newEvent Javascript function

The format of the function newEvent (which is called when you get an event), is as follows:

**newEvent()**

Request Argument

| Name | Type | Description |
|---|---|---|
| **`event`** | CEvent object | An object that encapsulates all the data for the event from the Message Bus, and allows you to forward the event to the bus, using the CEvent forward method detailed below. |

**CEvent forward methods**

You can emulate MoogDb behavior by running the MoogDb.V2 Moobots.For example, the **`alert.forward(this) line`** will send an alert onto the Moolets specified in the appropriate **`process_output_of`** block within **`moog_farmd.conf`**.

You could instead remove the **`process_output_of`** lines from the AlertRulesEngine / Sigaliser / Cookbook / Speedbird Moolets and explicitly send events / alerts / Situations on within the Moobot code using (as an example):

```
alert.forward("Cookbook");
```

The advantage of this approach is that alerts / Situations can be forwarded to different AlertRulesEngines / Sigalisers dynamically in the Moobots (for example based on the value of the source file).

### CEventAdaptor auxiliary object

This object is a utility class used by the Events module to allow for the programmatic activation of event listening. It has one method:

### listen()

Starts the event adaptor listening, which then calls the specified function when an event occurs.

Request Argument

None.

Return Parameter

Void - no value returned.

### CEvent auxiliary object

This object encapsulates a generic Message Bus event object, and the contents of it are specific to the event type it represents. You can however access the key-value pairs contained in the object, and also set the values. Its methods include:

### contains()

Returns true if the Event contains a value stored at the key **`name`**.

Request Argument

| Name | Type | Description |
|---|---|---|
| **`name`** | String | The name of the key being queried. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | True if the event has a field called **`name`**, otherwise false. |

### set()

Associates the specified **`value`** with the specified **`name`** in the event.

Previous **`key`** mapping has the old **`value`** replaced.

Request Argument

| Name | Type | Description |
|---|---|---|
| **`name`** | String | The key with which the specified value is to be associated. |
| **`value`** | Object | The value associated with the key. |

Return Parameter

| Type | Description |
|---|---|
| **`Boolean`** | Indicates if the operation was successful: true = success, false = fail |

### value()

Returns the object stored at the key **`name`**.

Request Argument

| Name | Type | Description |
|---|---|---|
| **`name`** | String | The name of the key to return the object from. |

Return Parameter

| Type | Description |
|---|---|
| **`Object`** | A Javascript object containing what is at the key **`name`**. |

Note:

Compatibility with MoogDb and MoogDb V2 Methods and auxiliary objects listed here are compatible with the MoogDb V2 module.

### CEvents API

Cisco Crosswork Situation Manager uses the CEvents API to pass data to LAMbot functions.

The methods are as follows:

### contains()

Check to see whether the payload of the CEvent contains the given key.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **`nm`** | String | The name of a potential key in the payload. |

Return Parameter

| Type | Description |
|---|---|
| boolean | Returns true if the provided key was found in the payload, or false if it was not. |

### evaluateFilter()

Allow an event/alert/Situation to be easily evaluated against a filter.

Request Arguments

| Name | Type | Description |
|---|---|---|
| Filter | String | A JSON or SQL-like filter for events, alerts or Situations. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Whether the filter matches the event, alert or Situation.<br><br>Returns true if the filter matches the event, alert or Situation.<br><br>Returns false if the filter has a correct syntax but doesn't match the event, alert or Situation.<br><br>Returns null if the filter syntax is incorrect. |

Example

```
var is_matching = situation.evaluateFilter("description LIKE 'Created Situation'");
```

forward(moobot)

This will forward the CEvent down the chain configured in the **`moog_farmd.conf`** (using **`process_output_of`** configuration), usual way of calling this is CEvent.forward(this) where this is the Moobot that's processing the CEvent object.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **`moobot`** | NativeObject | The instance of the Moobot which is handling the CEvent object. |

Return Parameter

None.

forward(target)

### forward(target,...)

Takes any number of target moolets names as strings and forwards the CEvent to each of them. For example CEvent.forward("moolet1") or CEvent.forward("moolet1", "moolet2").

Request Arguments

| Name | Type | Description |
|---|---|---|
| **targets** | Stringvarargs | Any number of String Moolet names which the CEvent is going to be forwarded to. |

Return Parameter

None.

### getActionDetails()

A utility helper method provided to retrieve the entire alert or Situation contained in the payload of a CEvent.

Request Arguments

None

Return Parameter

| Type | Description |
|---|---|
| JS NativeObject | The whole of the alert or Situation contained in the payload of the CEvent, as a NativeObject ready for use in the Javascript for a Moobot. |

### getCorrelationInfo()

Returns the correlation info for a Situation, which lists all of the services which are interested in this Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **replacementValue** | String/integer/boolean/object/map | The string or string/integer/boolean/object/map value to replace the value stored in the custom_info field. |

Return Parameter

| Type | Description |
|---|---|
| JS NativeObject | An object which contains the sig_id, service_name, external_id and properties for all the correlation info for the sig. sig_correlation_info is a one to many relationship of sigs to services. |

### getCustomInfo()

A helper method provided to retrieve the whole custom info object for an alert or Situation.

Request Arguments

None

Return Parameter

| Type | Description |
|---|---|
| JS NativeObject | The whole custom info map for an alert or Situation as a NativeObject ready for use in the Javascript for a Moobot. |

### Bot.getType()

Return the Moolet type

Request Arguments

None.

Return Parameter

| Type | Description |
|---|---|
| Enumerated type | Can be one of the following: |

Examples

After de-assigning a Situation, the previous moderator_id and status are displayed.

```
{moderator_id=2, last_state_change=1537794561, status=1}
```

After resolving a Situation, the previous status 4 (acknowledged) is displayed.

```
{last_state_change=1537867302, status=4}
```

### getSummaryData()

Fetches a summary of information about a system, such as the number of alerts or the service count bundled up as key/value pairs.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **replacementValue** | String/integer/boolean/object/map | The string or string/integer/boolean/object/map value to replace the value stored in the custom_info field. |

Return Parameter

| Type | Description |
|---|---|
| JS NativeObject | The summary of information about a system:<br>summary.alert_count - number<br>summary.service_count - number<br>summary.sig_summaries - map (contains "categories" and "queues")<br>summary.sig_summaries.categories - (array of objects)<br>summary.sig_summaries.queues - (array of objects)<br>categories and queues contain the following:<br>sig_total - number, alert_total - number, name - string<br>summary.sigs_down - number<br>summary.sigs_up - number<br>summary.total_events - number<br>summary.total_sigs - number |

### getTopic()

Returns the topic that the data was received on, for example "alerts" or "Situations".

Request Arguments

None

Return Parameter

| Type | Description |
|---|---|
| String | The name of the topic that the data came from or relates to, e.g. "Situations" or "alerts". |

### payload()

Retrieves the whole data payload that was sent in the CEvent object. In most cases the data contained in the payload is going to represent either a Situation or an alert, and as such will have key/value pairs which match the data columns for each.

Request Arguments

None.

Return Parameter

| Type | Description |
|---|---|
| CMooMsg | Enum value specifying the type of data that the Event contains and/or which topic the data was received on from the bus. |

Example

Example CEvent payload request:

```
logger.warning(cevent.payload().getData());
```

Example CEvent payload response:

```
{active=true, competencies=[], contact_num=, department=null,
description=Online, email=, fullname=cyber, groupname=End-User,
invitations=[], joined=1516963803, only_ldap=0, photo=-1, primary_group=1,
profile_image=null, realms=[DB], roles=[1, 3, 4, 5], session_expiry=null,
status=1, teams=[], timezone=SYSTEM, uid=6, username=cyber}
```

### set()

Used to insert or update a value in the payload of the CEvent.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **nm** | String | The key to insert or change a value at. |
| **val** | Object | The new value to store against the key. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Whether or not the value was successfully changed. |

### setCustomInfo()

Set or update the whole custom info object for an alert or Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **customInfo** | Js NativeObject | Thewhole custom info objectto set for an alert or Situation. |

Return Parameter

None.

### setCustomInfoValue()

Updates the custom information in the database for the specified Situation or alert.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **field** | String | The dot-formatted field within the custom_info of the reference alert or Situation to update. |
| **replacementValue** | String/integer/boolean/object/map | The string or string/integer/boolean/object/map value to replace the value stored in the custom_info field. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### setTopic()

Set or update the topic value in the payload of the CEvent object.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **topic** | String | The name of a topic to set or update in the payload data. |

Return Parameter

None

### stringValue()

Fetch a value from inside the payload which matches the provided key as a string value.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **name** | String | The key for a value stored in the payload which will be used to fetch the data. |

Return Parameter

| Type | Description |
|---|---|
| String | The value from the payload that was stored alongside the key (or null if no value was found to for the provided key) which has been converted to string format. |

### type()

Retrieves the type stored on the CEvent, this value indicates type of information in the payload and/or which topic the data came from.

Request Arguments

None.

Return Parameter

| Type | Description |
|---|---|
| EBotEvent | Enum value specifying the type of data that the Event contains and/or which topic the data was received on from the bus. |

### value()

Fetch a value from inside the payload which matches the provided key.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **nm** | String | The key for a value stored in the payload which will be used to fetch the data. |

Return Parameter

| Type | Description |
|---|---|
| Object | The value from the payload that was stored alongside the key (or null if no value was found to for the provided key) as an Object. |

Common fields for both Situations and alerts in the payload are:

- custom_info
- description
- first_event_time
- last_event_time
- last_state_change

## Events (MoogDb Only)

Compatibility with MoogDb and MoogDb.V2

Methods and auxiliary objects listed here are compatible with the MoogDb module, which was removed in v4.1.14.

Information here is provided for reference only.

For methods and auxiliary objects compatible with its replacement, see the MoogDb V2 module.

Description

The events Moobot module allows you to make a Moobot driven by the occurrence of events by defining the type of event that interests the Moobot.

The events module is available to load into any standard Moobot.

To use, at the top of a Moobot js file, define a new global object **events** to load the events module:

```
var events = MooBot.loadModule('Events');
```

Method

### events.onEvent

The events module has only one method, **onEvent**. This method points the Moobot to a supplied JavaScript function, which is called when a specified event type occurs.

The parameters to the called function depend on the type of event that you are listening for. In a Moobot, this method is typically the last line in a script.

The type of event adaptor chosen is specific to the type of Moobot you are building.

## Reference Guide

### events.onEvent()

Takes the name of a valid JavaScript function in a Moobot and also event code (from the constants module **eventType**), and returns an event adaptor object.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **functionName** | String | The name of a valid JavaScript function in the Moobot that is called when the event arrives. |
| **type** | CEvent | **eventType** event code that specifies what type of event the Moobot is listening for. It is typically from the constants module. |

Return Parameter

| Name | Type | Description |
|---|---|---|
| **CEventAdaptor** | Object | An event adaptor object. Made active with the listen function in-line to listen for the event type. |

Example

For the AlertBuilder MooBot:

```
events.onEvent("newEvent",constants.eventType("Event")).listen();
```

- Call the newEvent JavaScript function.
- Define the event type **event** (from the Constants module), which responds to events put on the Message Bus by a LAM.
- Call the listenfunction in-line to listen for the event type.

When the Moolet starts and loads this events Moobot, its JavaScript file executes, initialising the Moobot to respond in an event-driven way to events arriving.

newEvent Javascript function

The format of the function **newEvent** (which is called when you get an event), is as follows:

function newEvent()

Request Arguments

| Name | Type | Description |
|---|---|---|
| **event** | CEvent object | An object that encapsulates all the data for the event from the Message Bus. |
| **response** | CResponse object | An object to communicate back to the Moolet. The Moolet uses this response to broadcast any updates, or any changes to the data structures on the Message Bus. |

### CEventAdaptor auxiliary object

This object is a utility class used by the events module to allow for the programmatic activation of event listening. It has one method:

listen

### listen()

Starts the event adaptor listening, which then calls the specified function when an event occurs.

Request Argument

None.

Return Parameter

Void - no value returned.

### CEvent auxiliary object

This object encapsulates a generic Message Bus event object, and the contents of it are specific to the event type it represents. You can however access the key-value pairs contained in the object, and also set the values. Its methods include:

### contains()

Returns true if the event contains a value stored at the key name.

Request Argument

| Name | Type | Description |
|---|---|---|
| **name** | String | The name of the key being queried. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | True if the Event has a field called **name**, otherwise false. |

### set()

Associates the specified **value** with the specified **name** in the event. Previous key mapping has the old **value** replaced.

Request Argument

| Name | Type | Description |
|---|---|---|
| **name** | String | The key with which the specified value is to be associated. |
| **value** | Object | The value associated with the key. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail. |

### value()

Returns the object stored at the key **name**.

Request Argument

| Name | Type | Description |
|---|---|---|
| **name** | String | The name of the key to return the object from. |

Return Parameter

| Type | Description |
|---|---|
| Object | A Javascript object containing what is at the key name. |

### CEvent auxiliary object

Note

The following methods only apply to the MoogDb module, which is being deprecated.

### getJournalDetails()

Returns the details (if any) of the journaled operation for a Situation.

Request Argument

| Name | Type | Description |
|---|---|---|
| **scope** | Javascript object | The Moobot context, provided by using **this** as a parameter. |

Return Parameter

| Type | Description |
|---|---|

| | |
|---|---|
| Object | Javascript object containing the details of the journaled operation for a Situation. |

getCustomInfo

### getCustomInfo()

Returns the custom information (if any) for an alert or Situation.

Request Argument

| Name | Type | Description |
|---|---|---|
| **scope** | Javascript object | The Moobot context, provided by using this as a parameter. |

Return Parameter

| Type | Description |
|---|---|
| Object | Javascript object containing the custom information. |

### setCustomInfo()

Sets the custom information for an Alert or Situation

Request Arguments

| Name | Type | Description |
|---|---|---|
| **scope** | Javascript object | The Moobot context, provided by using **this** as a parameter. |
| **customInfoJS** | Native object | A Javascript object containing the custom information. |

Return Parameter

Void - no value returned.

### getCorrelationInfo()

Returns the external service **correlation_info** (where this has been set) for a Situation.

Request Argument

| Name | Type | Description |
|---|---|---|
| **scope** | Javascript object | The Moobot context, provided by using **this** as a parameter. |

Return Parameter

| Type | Description |
|---|---|
| Object | Javascript object containing the **correlation_info.** |

### getSummaryData()

Returns the summary information from a statistics summary event.

Request Argument

| Name | Type | Description |
|---|---|---|
| **scope** | Javascript object | The Moobot context, provided by using **this** as a parameter. |

Return Parameter

| Type | Description |
|---|---|
| Object | Javascript object containing the summary information. |

### CResponse auxiliary object

Note:

The following methods only apply to the MoogDb module, which has been deprecated.

**message()**

Object to broadcast on.

Request Argument

| Name | Type | Description |
|---|---|---|
| msg | CEvent | Object to broadcast on. |

Return Parameter

Void - no value returned.

**topic()**

Topic to broadcast message on.

Request Argument

| Name | Type | Description |
|---|---|---|
| **topic** | String | The topic name. |

Return Parameter

Void - no value returned.

**output()**

Freeform message to attach.

Request Argument

| Name | Type | Description |
|---|---|---|
| **txt** | String | The message as a text string. |

Return Parameter

Void - no value returned.

**retcode()**

The **retcode** value must be >= 0 for a message to be sent.

Request Argument

| Name | Type | Description |
|---|---|---|
| **code** | Number | Must be >= 0 for a message to be sent. |

Return Parameter

Void - no value returned.

doNotPropagate()

Indicates that no propagation is needed.

Request Argument

None.

Return Parameter

Void - no value returned.

## Expose Active Moolets

You can expose which Moolets are running by adding functions to a Moobot. The functions are:

- Bot.isActive: Returns whether the specified Moolet is active or not.
- Bot.getActiveMoolets: Returns a list of all active Moolets in the system.

### isActive

Returns whether the specified Moolet is active or not.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| **<mooletName>** | String | Yes | Name of a Moolet. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | 'true' indicates the Moolet is active, 'false' indicates it is inactive. |

Example

For example, you could use the function to return a logger warning if the ServiceNow Moolet is not running:

```
if(Bot.isActive('ServiceNow'))
        {
                var inform = mooletInforms.create('ServiceNow');
                inform.setSubject("ticket");
                inform.setDetails({sig_id: sigId}); inform.send();
        }
else
        {
                logger.warning("ServiceNow is not running - situation " +
sigId + " was not sent");
        }
```

### getActiveMoolets

Returns a list of all active Moolets in the Cisco Crosswork Situation Manager system.

Request Argument

None.

Return Parameter

| Type | Description |
|---|---|
| List | A list of all active Moolets in the Cisco Crosswork Situation Manager system. |

Example

You could use the function to return which Moolets are running if a specified Alert Workflow Engine Moolet is active:

```
var alert = moogdb.createAlert(event);
if(alert)
        {
                logger.info("New Alert Id: " + alert.value("alert_id"));
                if(Bot.isActive('AlertWorkflows'))
                {
                        logger.warning("Moolets running are: \n" +
Bot.getActiveMoolets());
                }
        }
```

An example log might return as follows:

```
WARN : [3:AlertBuilder][20190301 19:05:20.808 +0000] [AlertBuilder.js:128]
+|Moolets running are: [MaintenanceWindowManager, TeamsMgr, AlertBuilder,
SituationWorkflows, Housekeeper, Default Cookbook, Indexer,
```

```
EnrichmentWorkflows, AlertWorkflows, EventWorkflows, SituationMgr,
SituationRootCause]|+
```

## ExternalDb

### Description

The ExternalDb Moobot module allows Cisco Crosswork Situation Manager to access the following external relational databases (as well as any relational database that supports JDBC):

- MySQL
- Microsoft SQL Server
- IBM DB2
- Oracle
- PostgreSQL

With ExternalDb, Cisco Crosswork Situation Manager can retrieve information from external databases for use in alerts and Situations and can also update information in external databases with information from Cisco Crosswork Situation Manager.

ExternalDb is available to load into any standard Moobot.

To use, at the top of a Moobot js file, define a new global object **`externalDb`** to load the ExternalDb module:

```
var externalDb = MooBot.loadModule('ExternalDb');
```

**`externalDb`** is the name of the database.

### Reference Guide

#### externalDb.connect()

Establishes connection to an external database with defined connection properties.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **`<properties>`** | Object | A Javascript object containing connection properties. See below. |

Database connection properties

The ExternalDb module **`connect`** method defines connection properties as a Javascript object, which may include the following keys:

| Key | Description |
|---|---|
| **`type`** | The type of the database.<br><br>If type is omitted you must specify the URL, jar files and JDBC class name. To use an external database other than those in the supported list, omit the type from the connection properties. |
| **`host`** | The database host name or IP address (default is: 'localhost'). |
| **`database`** | The database name. |
| **`port`** | The port number. Default values:<br><br>MySQL - 3306<br><br>SQL Server - 1433 |

| | |
|---|---|
| | DB2 - 50000<br><br>Oracle - 1521<br><br>PostgreSQL - 5432 |
| **user** | The user name. If omitted can be specified in the URL (for some databases) or the properties. |
| **password** | The password. If omitted, it can be specified in the URL (for some databases) or the properties. |
| **encrypted_password** | Encrypted version of password (encrypted using **moog_encryptor**). |
| **properties** | A map of key-value pairs of properties to specify the connection properties. For example, **loginTimeout** for SQL Server or **useCompression** for MySQL. |
| **jar_files** | A list of the files locations indicating the the JDBC driver jar file location.<br><br>Defaults:<br><br>SQL Server - sqljdbc4.jar<br><br>DB2 - db2jcc4.jar<br><br>Oracle - ojdbc6.jar<br><br>PostgreSql - postgresql-9.3-1102.jdbc41.jar<br><br>Assumes it will find these files in **$MOOGSOFT_HOME/lib/cots/**<br><br>They are not bundled in a regular Moogsoft AIOps installation. |
| **class_name** | The name of the JDBC class. Defaults:<br><br>SQL Server - com.microsoft.sqlserver.jdbc.SQLServerDriver<br><br>MySQL - com.mysql.jdbc.Driver<br><br>DB2 - com.ibm.db2.jcc.DB2Driver<br><br>Oracle - oracle.jdbc.OracleDriver<br><br>PostgreSql - org.postgresql.Driver |
| **URL** | JDBC specific URL. If specified, it can override other properties. |
| **pool_properties** | A map of key-value pairs of properties of the connection pool that will be created. It may be used to define the number of connections made available to the external database by including the **pool_size** key.<br><br>**pool_size** The number of connections in the pool. Must be 1 or more, defaults to 10. Generally, this should match the number of threads configured to run the Moobot. |

Note:

You can also define connection properties in the following configuration file:
**moog_external_db_details.conf**

Return Parameter

| Type | Description |
|---|---|
| Object | A Java object containing connection details, depending on the requested connection properties<br><br>Returns **null** if no connection is available (due to either misconfiguration or unavailability of the external |

| | |
|---|---|
| | database). |

Note:

The **`connect`** method can accept a single parameter with connection properties, or two parameters - one with the generic connection properties and one specific for this connection. For example:

- **`var customersConnection = externalDb.connect(dbTypes.customers);`**

  will connect to the customer database as is.

- **`var customersConnection = externalDb.connect(dbTypes.customers, {user: 'admin', password: 'wrdPass'});`**

  will connect to the same database as the 'admin' user, with the password supplied.

You can also use the name from the following configuration file: **`moog_external_db_details.conf`**

Before making a connection, make sure the relevant database JDBC connector jar(s) are located where the configuration indicates. These are usually available for download from the database vendor.

Using the database connection:

The connection variable is a virtual connection, with the actual connections held and managed within the Java Virtual Machine. Therefore, there is no need to manage the connection, just call the connect method before you need to use the actual connection.

### externalDb.execute()

Performs an SQL update to the database.

The **`execute`** method has one string argument:

Request Argument

| Name | Type | Description |
|---|---|---|
| **`<argument>`** | String | SQL string argument. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail. |

Example

```
employeesConnection.execute('Update pets set species="dog" where species null');
```

### externalDb.query()

Performs an SQL query on the external database.

The **`query`** method has one string argument:

Request Argument

| Name | Type | Description |
|---|---|---|
| **`argument`** | String | SQL string argument. |

Return Parameter

| Type | Description |
|---|---|
| Object | A Java object which can contain the sub methods listed below<br><br>Returns **`null`** if the query fails. |

Response Methods

| Name | Description |
|---|---|
| **rows()** | Return the number of rows. |
| **next()** | Return the next row. |
| **rewind()** | Go back to the first row. |
| **hasNext()** | Indicate whether the current row is the last one. |
| **row(i)** | Return row **i** (zero based index). |
| **first()** | Return the first row. |
| **type(name)** | Return the type of column called **name** (or **null** if no such column exists). |
| **columnName(i)** | Return the name of column **i** (zero based index). |
| **isNumber(name)** | Returns true if the column name is a numeric column. |
| **isString(name)** | Returns true if the column name is a not numeric. |

Row Methods

| Name | Description |
|---|---|
| **value(name)** | Return the value for column named name as a string. |
| **columns()** | Return the number of columns. |
| **rewind()** | Go back to the first column. |
| **hasNext()** | Indicate whether the current column is the last one. |
| **next()** | Return the value in the next column as a string. |
| **column(i)** | Return the value in column **i** as a string. |
| **first()** | Return the value in the first column as a string. |
| **last()** | Return the value in the last column as a string. |

Example

```
var customers = customersConnection.query('Select * from customers');
while(customers.hasNext()==true)
{
    var customer=customers.next();
    var firstName = customer.value("first_name");
    var lastName = customer.value("last_name");
    logger.info(firstName + " " + lastName +" is a customer");
}
```

### externalDb.prepare()

Perform more complicated SQL queries or updates.

For example, you may need to reuse the same SQL statement with different arguments more than once, or you may need to use external data within the statement (and want to avoid SQL injection).

The **prepare** method has one string argument, where **?** can be used to define parameters within the SQL.

Request Argument

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **<argument>** | String | SQL string argument. |

Return Parameter

| Type | Description |
|---|---|
| Object | A prepared SQL statement object which can contain the following sub-methods listed below. |

Response Methods

| Name | Description |
|---|---|
| **set(i, value)** | Set parameter **i** (1 based index) to a value. Returns false in case of a failure. |
| **bind(value1, value2, value3,...)** | Set parameter 1 to value 1, parameter 2 to value 2 and so on. Returns false in case of a failure in either one. |
| **bindCount()** | Return the number of parameters needed to bind. Some vendors might not support this method in all cases, in case it is not supported '-1' is returned. |
| **execute(value1, value2, value3,...)** | Set parameter 1 to value 1, parameter 2 to value 2 and so on, and then execute the prepared statement. Returns false in case of a failure in one of the stages. If value are omitted will use the previously set or bind. |
| **query(value1, value2, value3,...)** | Set parameter 1 to value 1, parameter 2 to value 2 and so on, and then perform the query with the prepared statement. Returns **null** in case of a failure in one of the stages. Returns a Result Set (as the one in query above) if the operation was successful. If values are omitted, use the previously set or bind. |
| **close()** | Close the prepared statement.<br>Note:<br>It is important to close the statement with this method when no longer needed. |

Example

The following will set all pet species to "dog" if the breed is one of a specific dog breed:

```
var petsChange = employeesConnection.prepare('Update pets set species=?
where breed = ?');
petsChange.set(1, 'dog');
for (var breed in ['Labrador', 'Terrier', 'Beagle', 'Boxer', 'Poodle'])
{
    petsChange.set(2, breed);
    petsChange.execute();
}
petsChange.close();
```

[empty]

## Database Specific Information

Downloading JDBC Drivers

Note:

1. Be sure to download the correct version of the JDBC Driver for your database.
2. Ensure downloaded JDBC drivers are moved/copied to the $MOOGSOFT_HOME/lib/cots directory.

Microsoft SQL Server

JDBC driver: http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774

Connection properties: http://technet.microsoft.com/en-us/library/ms378672(v=sql.110).aspx

Example declarations:

```
testdb:
{
        type: 'sqlServer',
        host: '172.16.87.248',
        port: '1433',
        database: 'moog',
        user: 'sa',
        password: 'password'
}
```

or:

```
testdb:
{
        jar_files: ["/usr/share/moogsoft/lib/cots/sqljdbc4.jar"],
        class_name: "com.microsoft.sqlserver.jdbc.SQLServerDriver",
        url: "jdbc:sqlserver://172.16.87.248:1433;databaseName=moog",
        properties: { user: "sa", password: "password" }
}
```

MySQL

JDBC Driver: Already included in Cisco Crosswork Situation Manager - no need to download.

Connection properties: http://dev.mysql.com/doc/connector-j/en/connector-j-reference-configuration-properties.html

Example declarations:

```
testdb:
{
        type: 'mySql',
        host: '172.16.87.247',
        port: '3306',
        database: 'moog',
        user: 'root',
        password: 'm00gsoft'
}
```

or:

```
testdb:
{
        jar_files: ["/usr/share/moogsoft/lib/cots/mysql-connector-java-
5.1.37-bin.jar"],
        class_name: "com.mysql.jdbc.Driver",
        url: "jdbc:mysql://172.16.87.247:3306/moog",
        properties: { user: "root", password: "m00gsoft" }
}
```

IBM DB2

JDBC Driver: http://www-01.ibm.com/support/docview.wss?uid=swg21363866

Connection properties: http://www-01.ibm.com/support/knowledgecenter/SSEPGG_9.1.0/com.ibm.db2.udb.apdv.java.doc/doc/tjvjcccn.htm?cp=SSEPGG_9.1.0%2F8-1-4-2-1-0

Example declarations:

```
testdb:
{
        type: 'db2',
        host: '172.16.87.248',
        port: '50000',
        database: 'moog',
```

```
        user: 'db2admin',
        password: 'm00gsoft'
}
```

or:

```
testdb:
{
        jar_files: ["/usr/share/moogsoft/lib/cots/db2jcc4.jar"],
        class_name: "com.ibm.db2.jcc.DB2Driver",
        url: "jdbc:db2://172.16.87.248:50000/moog",
        properties: { user: "db2admin", password: "m00gsoft" }
}
```

Oracle

JDBC Driver: http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html

Connection properties: http://docs.oracle.com/cd/B28359_01/java.111/b31224/urls.htm

Example declarations:

```
testdb:
{
        type: 'oracle',
        host: '172.16.87.248',
        port: '1521',
        database: 'moog',
        user: 'System',
        password: '2pass'
}
```

or:

```
testdb:
{
        jar_files: ["/usr/share/moogsoft/lib/cots/ojdbc6.jar"],
        class_name: "oracle.jdbc.OracleDriver",
        url: "jdbc:oracle:thin:System/m00gsoft@172.16.87.248:1521:moog"
}
```

PostgreSql

JDBC Driver: https://jdbc.postgresql.org/download.html

Connection properties: http://jdbc.postgresql.org/documentation/head/connect.html

Example declarations:

```
testdb:
{
        type: 'postgresql',
        host: '172.16.87.248',
        port: '5432',
        database: 'moog',
        user: 'anotherUser',
        password: 'password'
}
```

or:

```
testdb:
{
        jar_files: ["/usr/share/moogsoft/lib/cots/postgresql-9.3-
1102.jdbc41.jar"],
```

```
        class_name: "org.postgresql.Driver",
        url: "jdbc:postgresql://172.16.87.248:5432/moog",
        properties: { user: "anotherUser", password: "password" }
}
```

## Graph Topology

The Graph Topology module uses an alternative clustering technique to refine accuracy and reliability, by using a shortest path measurement for clustering in Cookbook Moobot recipes.

In Cisco Crosswork Situation Manager, Situations can be generated by clustering events based upon the proximity in a network of the source devices sending the events. To do this, the source devices and their weighted connections are mapped in a topology.

Source devices in a network are represented as points in the topology called 'nodes'. Connections between the source devices are represented in the topology as 'edges'. Edges can be weighted to represent the connection length. If no weight is defined for an edge, the default value is 1. The number of connections on a node is called the 'degree' of the node.

### Distance

'Distance' is the shortest path between two nodes via the weighted edges (using Dijkstra's algorithm. More information from Wikipedia).

Topology data for the Graph Topology module, which is imported into Cisco Crosswork Situation Manager using the **`topology_builder`** utility, is in a CSV (comma separated value) file. Each edge defined in the CSV file is treated as representing a bi-directional connection between the specified nodes.

Each entry in the file names the two nodes that are connected, and (optionally) the weighted edge number, in the following format:

**`<first node>, <second node>, <weighted edge number>`**

If no **`<weighted edge number>`** is included, the default value of 1 is used.

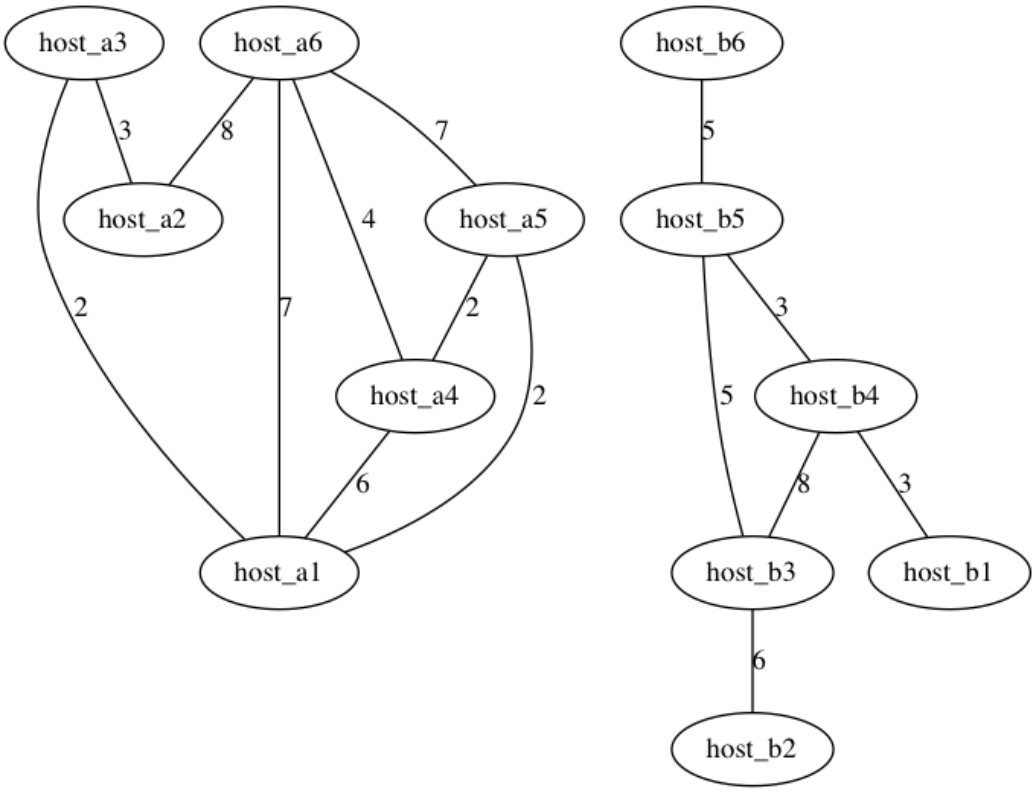
Example CSV file:

```
host_a3,host_a1,2
host_a3,host_a2,3
host_a4,host_a1,6
host_a5,host_a1,2
host_a5,host_a4,2
host_a6,host_a1,7
host_a6,host_a2,8
host_a6,host_a4,4
host_a6,host_a5,7
host_b3,host_b2,6
host_b4,host_b1,3
host_b4,host_b3,8
host_b5,host_b3,5
host_b5,host_b4,3
host_b6,host_b5,5
```

Note:

This data is used in the code examples below

The data above represents the following topology, with nodes named '**`host`**_...' and the weighted edges (see section on distances above) between them:




The Graph Topology module is available to load into any standard MooBot.

To use, at the top of a MooBot js file, define a new global object **`topo`** to load the Graph Topology module:

```
var topo = MooBot.loadModule('GraphTopo');
```

## Reference Guide

The Graph Topology module uses the following methods:

### topo.loadTopology()

Load the topology into the Graph Topology module and report success or failure. A failure to load may be because the **`topology_builder`** utility has not imported the topology data CSV file.

Request Argument

None.

Return Parameter

| Type | Description |
|---|---|
| Boolean | **`true`** = topology loaded successfully, **`false`** = topology failed to load |

Example

Request example to load a topology:

```
var ret = topo.loadTopology();
logger.warning("loadTopology -> " + ret);
```

Response if the topology loaded successfully:

```
WARN : ... [CLogModule.java]:99 +|loadTopology -> true|+
```

### topo.isConnected()

Check if a specified node is part of the topology.

Request Argument

| Name | Type | Description |
|---|---|---|
| **host** | String | The name of the node being checked |

Return Parameter

| Type | Description |
|---|---|
| Boolean | **true** = node in topology, **false** = node not in topology |

Examples

Using the example topology data above, running:

```
ret = topo.isConnected("host_a3");
logger.warning("isConnected 1 -> " + ret);

ret = topo.isConnected("does_not_exist");
logger.warning("isConnected 2 -> " + ret);
```

...returns the output below. The first node (**host_a3**) is in the topology, the second node (**does_not_exist**) is not:

```
WARN : ... [CLogModule.java]:99 +|isConnected 1 -> true|+

WARN : ... [CLogModule.java]:99 +|isConnected 2 -> false|+
```

**topo.connected()**

Check if there's a path between two specified nodes.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **host1** | String | The name of the first node being checked |
| **host2** | String | The name of the second node being checked |

Return Parameter

| Type | Description |
|---|---|
| Boolean | **true** = path between nodes exists, **false** = no path between nodes |

Examples

Using the example topology data above, running:

```
ret = topo.connected("host_a1", "host_a2");
logger.warning("connected 1 -> " + ret);

ret = topo.connected("host_a1", "host_b2");
logger.warning("connected 2 -> " + ret);
```

...returns the output below. The first path (between **host_a1** and **host_a2**) exists, second path (between **host_a1** and **host_b2**) does not:

```
WARN : ... [CLogModule.java]:99 +|connected 1 -> true|+

WARN : ... [CLogModule.java]:99 +|connected 2 -> false|+
```

**topo.distance()**

Check the Distance (shortest path) between two specified nodes, with an optional specified maximum Distance (**radius**).

Use **radius** to reduce the calculation time if you are not interested in long distances.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **host1** | String | The name of the first node being checked |
| **host2** | String | The name of the second node being checked |
| **radius** | number | Optional.<br>The maximum Distance to return a result for |

Return Parameter

| Type | Description |
|---|---|
| Number | The Distance between the two nodes. Returns **-1** if:<br>a node is not in the topology or the two nodes are not directly or indirectly connected<br>the Distance is larger than the (optionally) supplied **radius** |

Example 1

Using the example topology data above, run the following:

```
ret = topo.distance("host_a5", "host_a6");
logger.warning("distance 1 -> " + ret);
```

No radius is specified, so there is no maximum limit on the Distance (shortest path) returned.

All connections (direct and indirect) between nodes **host_a5** and **host_a6** are as follows:

| Edge value | Connection from **host_a6** to **host_a5** |
|---|---|
| 7 | Direct |
| 6 | via **host_a4** (2+4) |
| 9 | via **host_a1** (2+7) |
| 12 | via **host_a4** then **host_a1** (4+6+2) |
| 15 | via **host_a1** then **host_a4** (7+6+2) |
| 15 | via **host_a2** and **host_a3**, then **host_a1** (8+3+2+2) |
| 21 | via **host_a2** and **host_a3**, then **host_a1** and **host_a4** (8+3+2+6+2 |

```
WARN : ... [CLogModule.java]:99 +|distance 1 -> 6|+
```

The Distance (shortest path) between the nodes **host_a5** and **host_a6** is 6, and the output below is returned:

Note:

Although the direct connection between nodes **host_a5** and **host_a6** has an edge (weighted connection) of 7, the shortest path is the indirect connection via node **host_a4**, with a Distance of 6 (2 + 4)

Example 2

Using the example topology data above, run the following:

```
ret = topo.distance("host_b2", "host_b6", 8);
logger.warning("distance 2 -> " + ret);
```

The radius is specified as **8**. All connections (direct and indirect) between nodes **host_b2** and **host_b6** are as follows:

| Edge value | Connection from **host_2** to **host_b6** |
|---|---|
| 16 | via **host_b3** then **host_b5** (6+5+5) |
| 22 | via **host_b3**, then **host_b4** then **host_b5** (6+8+3+5) |

None of the connections have a path of 8 or less, so the result is **-1**, and the output below is returned:

```
WARN : ... [CLogModule.java]:99 +|distance 2 -> -1|+
```

Example 3

Using the example topology data above, running:

```
ret = topo.distance("host_a5", "host_b5");
logger.warning("distance 3 -> " + ret);
```

...returns the output below. The two nodes are not connected directly or indirectly, so **-1** is returned:

```
WARN : ... [CLogModule.java]:99 +|distance 3 -> -1|+
```

### topo.numberOfConnections()

Count the degree (number of connections) from a specified node.

Request Argument

| Name | Type | Description |
|---|---|---|
| **host** | String | The name of the node being checked. |

Return Parameter

| Type | Description |
|---|---|
| Number | The node's degree. Returns **0** if the node does not exist or has no connection. |

Example

Using the example topology data above, running:

```
ret = topo.numberOfConnections("host_b3");
logger.warning("numberOfConnections -> " + ret);
```

...returns the output below. The degree of node **host_b3** is 3:

```
WARN : ... [CLogModule.java]:99 +|numberOfConnections -> 3|+
```

### addEdge(String sourceNode, String sinkNode)

Optional parameter: Double weight (default value=1.0)

These will add a new node to a topology/graph both in memory and in the database.

Behavior:

- if unspecified, weight will have default value 1.0
- any new nodes will saved in memory and db
- new connection will be saved in memory and db

GraphTopo:

- won't work if there already is such edge

- uses jgraph methods addVertex and addEdge

Topo:

- won't work if both nodes aren't in topology or if both nodes already are in
- does not recalculate a topology, new coordinate == old coordinate + weight
- new coordinates will be saved in memory and database

## Logger

Warning:

The Logger module was deprecated for the release of Cisco Crosswork Situation Manager 7.1.0. See Configure Logging for details on the the new Logger.Configure Logging

The Logger module sets the log level in Moogfarmd, allowing log messages to be written to the common Moogfarmd log file. See Configure Logging for information on configuring logging.Configure Logging

For example, when you write a Moobot, you can use the **`Logger`** for debug. Writing a log message to a log file is an IO operation, and comes with execution time cost. When developing the Moobot it can be helpful to have a number of logging statements. Once the Moobot is operational, however, you should keep log messaging to a minimum.

The Logger module is available to load into any standard Moobot.

To use, at the top of a Moobot js file, define a new global object **`logger`** to load the Logger module:

```
var logger = MooBot.loadModule('Logger');
```

### Reference Guide

The **`logmessage`** argument used in the Logger module is a single string.

Multiple arguments are possible using concatenation. See Examples.

Note:

**`printf`** based Logger functions have been deprecated in favour of the 'single string argument' version. For more information click here.

### logger.debug()

Sends a debug log message (the lowest severity level). For example, this can be used for logging detailed troubleshooting information (not for production). See Examples.

Request Argument

| Name | Type | Description |
|---|---|---|
| **`logmessage`** | String | A single string of valid JavaScript variables or objects, used to form a log message. |

Return Parameter

Void - no value returned.

### logger.info()

Sends an information log message (the intermediate severity level). For example, this can be used to log the changing of a setting. See Examples.

Request Argument

| Name | Type | Description |
|---|---|---|
| **`logmessage`** | String | A single string of valid JavaScript variables or objects, used to form a log message. |

Return Parameter

Void - no value returned.

### logger.warning()

Sends a warning log message (a higher severity level). For example, this can be used to log behavior which impacts normal operation of the system. See Examples.

Request Argument

| Name | Type | Description |
|---|---|---|
| **logmessage** | String | A single string of valid JavaScript variables or objects, used to form a log message. |

Return Parameter

Void - no value returned.

### logger.fatal()

Sends a fatal log message (the highest severity setting). For example this can be used to log extreme circumstances, such as an unrecoverable failure that caused Moogfarmd to exit. See Examples.

Request Argument

| Name | Type | Description |
|---|---|---|
| **logmessage** | String | A single string of valid JavaScript variables or objects, used to form a log message. |

Return Parameter

Void - no value returned.

Examples

All the above methods work in the same way, with each sending a log message of a different severity level.

```
{
   var dispText= "Reset";
   var dispNum= 2;
   var aReal= 3.141593;
   var aString= "CPU@ >90%";
   var intHigh= 4;
   var intHighest= 5;

logger.debug("A debug message");

logger.info("Counter: "+ dispText);

logger.info("Severity low. Level: "+ dispNum + ". ...Pi = "+ aReal);

logger.warning("Warning: "+ aString);

logger.warning("Severity high. Level: "+ intHigh);

logger.fatal("Severity exceeds "+ intHighest + "! Restart required");
}
```

The above six logger arguments give the following six corresponding log messages:

```
DEBUG:... ...A debug message

INFO :... ...Counter: Reset

INFO :... ...Severity low. Level: 2. ...Pi = 3.141593
```

```
WARN :... ...Warning: CPU@ >90%

WARN :... ...Severity high. Level: 4

FATAL:... ...Severity exceeds 5! Restart required
```

## Mailer

The Mailer module allows you to send an email in response to events occurring in Cisco Crosswork Situation Manager.

You can load it into any standard Moobot. For example, you can load Mailer into Notifier.js Moobot and send users emails if they are invited to a Situation Room.

### Configure Mailer

To load the Mailer module, define a new global object mailer at the top of the Moobot JavaScript file:

```
var mailer = MooBot.loadModule('Mailer');
```

You can configure Mailer using the methods listed below.

### Methods

mailer.initTransport(mailerObj)

Defines the mail server information needed to send the email in the send function.

Request Argument

| Name | Type | Description |
|---|---|---|
| **mailerObj** | Object | A JSON object specifying connection properties |

Example

```
mailer.initTransport({
        server          : "smtp.mailserver.com",
        port            : 2525,
        account         : "user@mailserver.com",
        password        : "m00gsoft",
        isEncrypted     : false,
        start_tls       : false,
        use_tls         : false
        });
```

In general, use the guidelines below for the following ports:

- If using port 587, set **start_tls** to true and **use_tls** to false.
- If using port 465, set **start_tls** to true and **use_tls** to true.
- If using port 25, set **start_tls** to false and **use_tls** to false (or comment both flags out).

Note: If you do not want Mailer to send authentication credentials to the SMTP mail server, do not specify the **password** field:

```
mailer.initTransport({server: "yourhostname", port: 25,
account:"username@emailhost.com" });
```

If **password** is omitted, an unauthenticated connection is created between Mailer and the server.

**mailer.send(mailMsg)**

Use this method to send email. A callback function needs to be defined in the same Moobot and referenced in the mailMsg which is executed after a successful transmission.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **mailMsg** | Object | A JSON object containing fields needed to populate the email. |

Example

```
var mailMsg = {
        to      : "destination@mail.com",
        subject : "MOOGsoft Situation Room Notification",
        message : "email body",
        invite  : invite, // do not change
        bot     : MooBot.self, // do not change
        callback: "sendSuccess", // the name of the function to run in
this Moobot
        args    : [ invite_id, "Sent successfully",vector ] // do not
change
        };
mailer.send(mailMsg);
```

## MoogDb V2

You can query and manipulate a variety of entities in the Cisco Crosswork Situation Manager database using the MoogDb V2 Moobot module.

The module uses various methods to retrieve information from MoogDb and update components of Cisco Crosswork Situation Manager including alerts, Situations, users and teams.

All MoogDb V2 methods that update the database also publish information about the appropriate updated entities on the Configure the Message Bus, so any updated information automatically appears in Cisco Crosswork Situation Manager when the relevant method is called.Configure the Message Bus

### Load MoogDb V2

You can load the MoogDb V2 module into any standard MooBot by defining a new global object called **moogdb** at the top of the JavaScript file:

```
var moogdb = MooBot.loadModule('MoogDb.V2');
```

### Methods

All available MoogDb V2 methods are described in the sections below:

**addAlertToSituation**

Adds a specified Alert to a Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | The Alert ID |
| **situationId** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### addCorrelationInfo

Adds correlation information (external service name and external entity ID) to a Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **service** | String | The name of the external service, such as **ServiceNow** |
| **externalId** | String | The identifier that the entity has in the external service, which corresponds to the Situation |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### addSigCorrelationInfo

Adds correlation information (external service name and external entity ID) to a Situation. This is the recommended method for adding correlation information to a Situation, the **addCorrelationInfo** method has been retained for backwards compatibility.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **service** | String | The name of the external service, such as **ServiceNow** |
| **externalId** | String | The identifier that the entity has in the external service, which corresponds to the Situation |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### addProcess

Adds a new process to the database.

Processes are external business entities related to business activities that are affected by the incidents that Cisco Crosswork Situation Manager captures in Situations.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **process** | String | The process name |
| **description** | String | The process description. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### addService()

Adds a new external service to the database.

An external service is a business entity monitored by Cisco Crosswork Situation Manager via Event streams.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **service** | String | The name of the external service being added |
| **description** | String | The service description |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### addThreadEntry

A POST request that adds a new entry to an existing thread in a Situation. Optionally, you can specify the new entry as being a resolving step.

Threads are comments or 'story activity' on Situations.

This endpoint returns the entry ID of the newly created thread entry.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **entry** | String | Yes | Description of the new entry you want to add to the existing thread. For example, "And another thing...". |
| **thread_name** | String | Yes | Name of the existing thread. |
| **user_id** | Nunber | Yes | A valid user ID. |
| **sitn_id** | Number | Yes | Situation ID. |
| **resolving_step** | Boolean | No | Whether or not the thread entry you are adding is a resolving step. Defaults to **false** if not specified, |

Return Parameter

| Type | Description |
|---|---|
| HTTP code | HTTP status or error code indicating request success or failure.<br><br>For codes, see HTTP Status and Error Codes. |

Successful requests return a JSON object containing the following:

| Name | Type | Description |
|---|---|---|
| entry_id | Number | ID of the new thread entry. |

Examples

Request to add an entry "New Entry" to thread "Support" in Situation 158 using user ID 47. The resolving step parameter defaults to false.

```
var newThreadEntryID = moogdb.addThreadEntry(“New Entry”, “Support”, 47, 158);
```

Request to add an entry "New Entry" to thread "Support" in Situation 58 using user ID 47. This thread entry is a resolving step:

```
var newThreadEntryID = moogdb.addThreadEntry(“New Entry”, “Support”, 47, 158, true)
```

Returns the new thread entry ID:

```
345
```

### assignAlert

Assigns an Alert to a valid user, identified by their user ID.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | The Alert ID |
| **userId** | Number | A valid user ID |
| **username** | String | A valid user name |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### assignAndAcknowledgeAlert

Assigns an Alert to a valid user, identified by their user ID, and acknowledge the alert.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | The Alert ID |
| **userId** | Number | A valid user ID |
| **username** | String | A valid user name |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### assignAndAcknowledgeSituation

Assigns a Situation to a valid user, identified by their user ID, and acknowledge the situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **userId** | Integer | A valid user ID |
| **username** | String | A valid user name |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### assignModerator

Assigns a Situation to a valid user, identified by their user ID.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |

| | | |
|---|---|---|
| **moderatorId** | Number | A valid user ID |
| **username** | String | A valid user name |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### assignTeamsToSituation

Assigns one or more teams to a Situation. Once successfully run, Cisco Crosswork Situation Manager marks the Situation as overridden and the Teams Manager Moolet can no longer modify its team assignment. See Teams Manager Moolet for more information.

The method replaces any teams previously assigned to the Situation. You can also use it to unassign all teams from a Situation.

Request Arguments

Include either **team_ids** or **team_names**.

| Name | Type | Description |
|---|---|---|
| **sitn_id** | Number | The Situation ID. |
| **team_ids** | JSON list | A list of team IDs to assign to the Situation. Specify an empty list to unassign all teams from the Situation. |
| **team_names** | JSON list | A list of team names to assign to the Situation. Specify an empty list to unassign all teams from the Situation. |

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript object containing a list of the team names or team IDs assigned to the Situation, depending on the input. |

Input Example 1:

```
var assignTeamIDs = moogdb.assignTeamsToSituation(1, { "team_ids" : [1, 2] } )
```

Return:

```
{ "team_ids" : [1, 2] }
```

Input Example 2:

```
var assignTeamNames = moogdb.assignTeamsToSituation(2, { "team_names" : [ "Team1", "Team2" ] } )
```

Return:

```
{ "team_names" : [ "Team1", "Team2" ] }
```

Unassign Example:

```
var unassignTeamIDs = moogdb.assignTeamsToSituation(1, { "team_ids" : [] } )
```

Return:

```
{ "team_ids" : [] }
```

### closeAlert

Closes one or more alerts.

Request Argument

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | A single alert ID |
| **alertIds** | List | A list of alert IDs |
| **thread_entry_comment** | String | Optional comment |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

Example Input

```
var success = moogdb.closeAlert([78,234,737], "Closing as agreed during team discussion 1/1/2018");
```

### closeSituation

Closes a Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **closeAlerts** | Constant | Determines how the Alerts in the Situation are treated:<br>**CLOSE_NO_ALERT** - No Alerts are closed<br>**CLOSE_ALL_ALERTS** - All Alerts are closed<br>**CLOSE_UNUSED_ALERTS** - Only the Alerts unique to this Situation (i.e. otherwise unused) are closed<br>To access these constants from a MooBot, precede them with the module name, for example:<br>**moogdb.CLOSE_NO_ALERT** |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### createAlert

Creates or updates an Alert in the database. Optionally updates custom info for de-duplicated Alerts.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **alert** | Native object | A Javascript object containing Alert attributes, such as **type**, **severity**, etc |
| **event** | CEvent | A CEvent object representing the Alert, containing Alert attributes, such as **type**, **severity**, etc |
| **mergeCustomInfo** | Boolean | Set this to 'true' to merge the **custom_info** data in this Alert with the info held in the database<br>Optional |

Return Parameter

| Type | Description |
|---|---|
| **CEvent** | A CEvent object containing the latest version of the Alert |

### createMaintenanceWindow

Creates a maintenance window that filters alerts, by passing an object containing the information.

Request Arguments

| Name | Type | Description |
|---|---|---|
| maintenanceWindowObj | Object | A map containing the following information. |
| **name** | String | Mandatory - The name of the maintenance window. |
| **description** | String | Mandatory - The description of the maintenance window. |
| **filter** | String | Mandatory - The filter to apply to the new alerts created. |
| **start_date_time** | Number<br><br>(Epoch) | Mandatory - The time in epoch where the maintenance window will start, up to a maximum of 5 years in the future. |
| **duration** | Number<br><br>(seconds) | Mandatory - The duration in seconds where the maintenance window is running, must be greater than zero. |
| **forward_alerts** | boolean | Mandatory - Whether the alert will be forwarded to situation or not. |
| **recurring_period** | Number | Optional - Whether or not this is a recurring maintenance window. Set this to:<br><br>**1** for a recurring maintenance window.<br><br>**0** for a one-time maintenance window.<br><br>If you change this from **0** to **1**, you must specify **recurring_period_units**. |
| **recurring_period_units** | Number | Specifies the recurring period of the maintenance window, in days, weeks or months. If you set **recurring_period** to **0**, you must set **recurring_period_units** to **0**. Valid values are:<br><br>**0** = a one-time maintenance window<br><br>**2** = daily<br><br>**3** = weekly<br><br>**4** = monthly |

Input example :

```
{
    "name": "Mike",
    "description": "A description",
    "filter": "{'column': 'source', 'op': 0, 'value': '\'Nile\'', 'type': 'LEAF'}",
    "start_date_time": 1497971059,
    "duration": 360000,
    "forward_alerts": true,
    "recurring_period": 1,
    "recurring_period_units": 2
}
```

Return Parameter

| Type | Description |
|---|---|
| Long | The window ID created, or null if an error occurred. |

### createSituation

Creates a new Situation, containing no Alerts.

Situation settings, such as severity are defined in the following arguments:

Request Arguments

| Name | Type | Description |
|---|---|---|
| **moderator** | String | A valid user name |
| **label** | String | The new Situation description |

Return Parameter

| Type | Description |
|---|---|
| CEvent | The newly created Situation wrapped in a CEvent object |

### createTeam

Create a new team, by passing an object containing team information.

Request Arguments

| Name | Type | Description |
|---|---|---|
| teamObj | Object | A map containing the following parameters |
| **name** | String | Mandatory - the new team (unique) name |
| **alert_filter** | String | Optional - The team alerts filter. Either a SQL like filter or an JSON representation of the filter |
| **services** | JSON list | Optional - List of the team services names or IDs |
| **sig_filter** | String | Optional - The situation filters. Either a SQL like filter or an JSON representation of the filter |
| **landing_page** | String | Optional - The team default landing page |
| **active** | Boolean | Optional - False if the team is inactive, true if the team is active. Default to true |
| **description** | String | Optional - The team description |
| **users** | List of numbers or strings | Optional - The team users (either IDs or usernames) |

Input example :

```
{
        "name": "myTeam",
        "alert_filter": "{ \"column\": \"count\", \"op\": 1, \"value\": 1,
\"type\": \"LEAF\" }",
        "sig_filter": "{ \"column\": \"severity\", \"op\": 1, \"value\":
5, \"type\": \"LEAF\" }",
        "active": true,
        "services": [1, 2, 4],
        "users": ["user1", "user4"],
        "description": "myDescription",
        "landing_page": ""
}
```

Return Parameter

| Type | Description |
|---|---|
| Integer | The team id created, or null if an error occurred. |

### createThread

Creates a new thread for a Situation.

Threads are comments or 'story activity' on Situations.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **thread** | String | The name of the new thread |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### createThreadEntry

Note:

This method has been superseded. Use “addThreadEntry” instead. All new functionality will be delivered in addThreadEntry.

Creates an entry on the specified thread. This method returns a Boolean indicating whether or not the thread entry was created successfully.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **entry** | String | The entry as a text string |
| **thread** | String | The name of the thread |
| **userId** | Number | A valid user ID |
| **situationId** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### createUser

Create a user, by passing an object containing user properties

Request Arguments

| Name | Type | Description |
|---|---|---|
| **userObj** | Object | A map containing the following user information |
| **username** | String | Mandatory - the new user (unique) login username |
| **password** | String | The new user password (only valid for DB realm) |
| **active** | Boolean | true if the user active, false if the user inactive, default to true |

| | | |
|---|---|---|
| **email** | String | The user email address |
| **fullname** | String | The user full name |
| **roles** | JSON list | Mandatory - List of user roles. That list should contain either the list the role IDs or the role names. E.g "roles":["Super User"], |
| **primary_group** | String or Number | The user primary group name or primary group id |
| **department** | String or number | The user department id or name |
| **joined** | Number | The time the user joined (in Unix time) |
| **timezone** | String | The user timezone |
| **contact_num** | String | The user phone number |
| **session_expiry** | Number | The number of minutes after which the user session will expire. Default to system default |
| **competencies** | JSON list | A list with the user competencies. Each competency should have have name or cid and ranking. That is, something like: **[{"name":"SunOS", "ranking": 40},{"name":"SAP", "ranking": 50},{"name":"EMC", "ranking": 60}]** |
| **teams** | JSON list of numbers or strings | List of the user teams. The list should contains either the list of the teams ID or the teams name |

Example

Input:

```
{
        "username": "user1",
        "fullname": "firstName surName",
        "competencies": [{
                        "name": "SunOS",
                        "ranking": 40
                },
                {
                        "name": "SAP",
                        "ranking": 50
                },
                {
                        "name": "EMC",
                        "ranking": 60
                }
        ],
        "roles": ["Super User"],
        "department": 3,
        "active": true,
        "email": "user@email.com",
        "timezone": "a timezone",
        "teams": [1, 2, 4],
        "joined": 12345678,
        "contact_num": "0965412345"
}
```

Return Parameter

| Type | Description |
|---|---|

| | |
|---|---|
| Integer | The user id created, or null if an error occured |

**createWorkflow**

Create a new Workflow at the end of the Moolet sequence. To move it, use reorderWorkflow().

Request Arguments

| Name | Type | Description |
|---|---|---|
| moolet_name | String | |
| workflow_name | String | |
| description | String | Workflow description. |
| entry_filter | Filter in JSON or SQL format | The entry filter of the Workflow. Missing, null, or empty means "accept all." |
| sweep_up_filter | Filter in JSON or SQL format | Check the database for all objects that match the filter criteria and pass them to all workflow actions as a list parameter. The sweep-up filter expedites entry of related objects into the workflow.<br><br>For example if you receive a link-up alert, you can set a filter to retrieve all related link-down alerts from the database and have the sweep up filter close them.<br><br>A missing, null, or empty argument implies no sweep-up filter. |
| first_match_only | Boolean | If True, perform workflow operations once only on each object. |
| operations | JSON Array | Required<br><br>A list of operations, each has:<br><br>Type - (ENUM/String, Required) the type of the operation. Either action, decision, delay. Depending on the type, we'll read other fields in the JSON<br><br>Operation_name - (String, Required for action/decision type) Operation name.<br><br>function_name - (String, Required for action/decision type) Function name.<br><br>function_args - (JS Object , action/decision types, only optional) Function arguments.<br><br>Duration - (Integer, Required for delay type) The number of seconds before the message goes to the next operation/Workflow/moolet.<br><br>Reset - (Boolean, Required for delay type)Reset the timer on each occurrence? |

Example

```
var id = moogdb.createWorkflow(
{
    "moolet_name": "Alerts Workflows",
    "workflow_name": "ChangeInfoWorkflow",
    "description": "Changingthealertinformation",
    "entry_filter": {
            "column": "severity",
            "op": 5,
            "value": 3,
            "type": "LEAF"
    },
    "sweep_up_filter": {
            "column": "description",
            "op": 4,
            "value": "description",
            "type": "LEAF"
```

```
        },
        "first_match_only": false,
        "operations": [{
                        "type": "action",
                        "function_name": "functionA",
                        "function_args": {
                                "admin": 2
                        },
                        "operation_name": "do something"
                },
                {
                        "type": "delay",
                        "delay": 30,
                "reset": false
                }
        ]
});
```

### deAssignAlert

Deassigns an alert. Removes the user assigned to the alert and leaves it unassigned.

Request Argument

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | The alert ID |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### deleteMaintenanceWindow

Delete a single maintenance window.

Request Argument

| Name | Type | Description |
|---|---|---|
| **maintenanceWindowId** | Number | The maintenance window ID |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

Example

Request to delete maintenance window 456:

```
var success = moogdb.deleteMaintenanceWindow(456)
```

Successful return:

```
true
```

### deleteMaintenanceWindows

Delete maintenance windows that match the specified filter. The filter can be JSON or SQL (advanced). See “Filter Search Data” for further information on creating filters in Cisco Crosswork Situation Manager.

Request Argument

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **filter** | String | The filter to delete windows by. Something like: description matches 'maint_window_12' |
| **limit** | Number | The maximal number of windows to fetch. default to 100. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

Examples

Request to delete maintenance windows that match a filter:

```
var success = deleteMaintenanceWindows(filter, limit);
```

JSON filter where the description is "host375":

```
{ "column": "description", "op": 10, "value": "host375", "type": "LEAF" }
```

Advanced SQL filter where the description is "host375":

```
Description MATCHES "host375"
```

Successful return:

```
true
```

### deleteWorkflow()

Deletes a Workflow moolet.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **id** | Integer | Required -- ID of the workflow to delete. |

Response Parameter

| Type | Parameter |
|---|---|
| Boolean | True if the operation succeeded. |

### getActiveSituationIds

Returns the total number of active Situations, and a list of their Situation IDs. Active Situations are those that are not Closed, Resolved or Dormant.

Request Arguments

None. The above method returns data on all active Situations.

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript object containing the total and the Situation IDs |

Example

Return:

```
{
     "total_situations":10,
     "sitn_ids":[4, 5, 6, 12, 14, 15, 16, 17, 18, 19]
}
```

### getAlert

Fetches a specified alert from the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | The alert ID |

Return Parameter

| Type | Description |
|---|---|
| CEvent | A CEvent object containing the alert attributes, such as **type**, **severity**, etc. |

### getAlertActions

Retrieves the actions for one or more specified alerts or for a specified time period.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| alert_ids | JSON list | No | List of alert IDs. |
| start | Number | Yes | Starting row from which data should be included. |
| limit | Number | Yes | Maximum number of actions you want to retrieve. |
| actions | JSON list | No | List of action codes. If no action codes are specified, all action codes are returned. See Alert Action Codes for a list of action codes and their descriptions. Only action codes 8 (Alert Resolved) and 9 (Alert Closed) are valid. |
| from | Number | No | Start time (in Unix epoch time) of the period you want to retrieve alert actions for. |
| to | Number | No | End time (in Unix epoch time) of the period you want to retrieve alert actions for. |

Return Parameters

| Type | Description |
|---|---|
| Native object | A JSON object containing the alert action information. |

Examples

Request:

```
var actions = moogdb.getAlertActions(request);
```

Example **request** object to return the first 100 actions for alert IDs 1 and 2 for action codes 9 and 10:

```
{
  “alert_ids” : [1, 2],
  “start”: 0 ,
  “limit” : 100,
  “actions” : [8, 9]
}
```

Example **request** object to return the first 100 actions for alert IDs 1 and 2 for action codes 9 and 10 between the Unix epoch times 1553861746 and 1553872546:

```
{
            "alert_ids" : [1, 2],
            "limit" : 100,
            "actions" : [8, 9],
            "from" : 1553861746,
            "to" : 1553872546
}
```

Successful return:

```
[{
    "uid": 49,
    "action_code": 8,
    "description": "Alert Resolved",
    "details": {},
    "alert_id": 1,
    "timed_at": 1557504393
    }, {
    "uid": 49,
    "action_code": 9,
    "description": "Alert Closed",
    "details": {},
    "alert_id": 1,
    "timed_at": 1557504912
    }
}]
```

### getAlertCustomInfo

Retrieves any custom information from a specific alert.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | ID of the alert you want to retrieve custom info data from. |
| **key** | String | Specify the key if you are interested in a specific value. Otherwise the method returns all **custom_info** information. |

Return Parameter

| Type | Description |
|---|---|
| Number, List, String or Object | A map of name-value pairs containing the new **custom_info** information. |

### getAlertIds

Retrieves all alert IDs matching the query.

Request Argument

| Name | Type | Description |
|---|---|---|
| **query** | JSON Object | A JSON object containing the alert filter information. |
| **limit** | Number | The maximum number of alert ids to return. |

Return Parameter

| Type | Description |
|---|---|
| NativeObject | A Javascript object containing the total number of alerts and their alert IDs. |

Example

Return:

```
{
    "total_alerts":10,
    "alert_ids":[4, 5, 6, 12, 14, 15, 16, 17, 18, 19]
}
```

### getSigCorrelationInfo

Retrieves all correlation information related to a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **sitn_id** | Number | The Situation ID. |

Return Parameter

| Type | Description |
|---|---|
| Object | A Javascript object containing a list of maps of correlation info. |

### getSigCustomInfo

Retrieves all custom information related to a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **sigId** | Number | The Situation ID. |
| **key** | String | Node path for specific value to return. |

Return Parameter

| Type | Description |
|---|---|
| Number, object, list or string | Depends on the key but can either be a number, object, list or string containing a list of maps of custom info. |

### getMaintenanceWindows

Get all maintenance windows based on the window id and how many should be fetched.

Request Argument

| Name | Type | Description |
|---|---|---|
| **start** | Number | The start point for where to fetch windows from (ie, 0 to start at the first, 10 to start at the 11th) |
| **limit** | Number | The number of windows to fetch |

Return Parameter

| Type | Description |
|---|---|
| NativeObject | A Javascript object with a nested array. |

Example

Return:


```
{
  "windows": [
    {
      "filter":
"{\"op\":6,\"column\":\"severity\",\"type\":\"LEAF\",\"value\":[2]}",
      "duration": 3600,
      "recurring_period": 1,
      "del_flag": false,
      "forward_alerts": false,
      "last_updated": 1491917013,
      "name": "window1",
      "updated_by": 3,
      "description": "dfgvhbjk",
      "id": 1,
      "recurring_period_units": 2,
      "start_date_time": 1491916979
    }
  ]
}
```

### findMaintenanceWindows

Find maintenance windows based on a filter and how many should be fetched.

Request Argument

| Name | Type | Description |
|---|---|---|
| **filter** | String | The filter to find windows by. Something like: description matches 'dfgvhbjk'. |
| **limit** | Number | The maximal number of windows to fetch. default to 100. |

Return Parameter

| Type | Description |
|---|---|
| PublicObject | A Javascript object containing the windows |

Example

Return:


```
{
  "windows": [
    {
      "filter":
"{\"op\":6,\"column\":\"severity\",\"type\":\"LEAF\",\"value\":[2]}",
      "duration": 3600,
      "recurring_period": 1,
      "del_flag": false,
      "forward_alerts": false,
      "last_updated": 1491917013,
      "name": "window1",
      "updated_by": 3,
      "description": "dfgvhbjk",
      "id": 1,
      "recurring_period_units": 2,
      "start_date_time": 1491916979
    }
  ]
}
```


### getQueueName

Fetches the queue name from the database, for the given queue ID.

Request Argument

| Name | Type | Description |
|---|---|---|
| **queueId** | Number | The queue ID |

Return Parameter

| Type | Description |
|---|---|
| String | Queue name |

### getPrcLabels

Returns probable root cause (PRC) information for all alerts or specified alerts within a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **alert_ids** | JSON list | A list of the alert IDs (Optional) |

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript object containing the probable root cause information for the alerts in the specified Situation |

Example

Input:

```
var alertIds = [1,2,3,4];
var prcLabels = moogdb.getPrcLabels(1, alertIds);
```

Return:

```
{
      "non_causal":
        [2,3],
      "unlabelled":
        [4],
      "causal":
        [1]
}
```

### getProcesses

Fetches a list of processes from the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| **limit** | Integer | The maximum number of processes to retreive. 1000 is the default. |

Return Parameter

| Type | Description |
|---|---|
| Native object | A list of string hashmaps describing the requested processes, or null value if there is an error. |

### getResolvingThreadEntries

Returns thread entries that have been marked as resolving steps for the specified Situation. Threads are comments or 'story activity' on Situations.

You can select specific thread entries to return using start and limit values. If not, the first 100 entries will be returned. The entries returned are ordered by most recent entries first.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **situationId** | Number | Yes | Situation ID. |
| **thread** | String | Yes | Name of the thread. |
| **start** | Number | No | Number of the first thread entry to return. |
| **limit** | Number | No | Maximum number of thread entries to return. |

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript object containing details of the selected thread entries. |

Examples

Request to return the first 100 thread entries that are resolving steps for Situation 58:

```
var resolvingEntries = moogdb.getResolvingThreadEntries(58);
```

Request to return the first 10 thread entries that are resolving steps for Situation 58:

```
var resolvingEntries = moogdb.getResolvingThreadEntries(58, 0, 10);
```

Return:

```
{
  "entries": [
              {
                  "uid": 3,
                  "entry": "This one is important. Another comment",
                  "agrees": [],
                  "total_comments": 0,
                  "thread_id": "Support",
                  "mmid": -1,
                  "sig_id": 1,
                  "entry_id": 2,
                  "timed_at": 1423226829,
                  "disagrees": [],
                  "commenters": []
              },
              {
                  "uid": 3,
                  "entry": "No comment. A comment",
                  "agrees": [],
                  "total_comments": 0,
                  "thread_id": "Support",
                  "mmid": -1,
                  "sig_id": 1,
                  "entry_id": 1,
                  "timed_at": 1423226807,
                  "disagrees": [3],
                  "commenters": []
              }
             ],
  "total_entries": 2
}
```

### getServices

Fetches a list of services from the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| **limit** | Integer | The maximum number of services to retrieve. 1000 is the default. |

Return Parameter

| Type | Description |
|---|---|
| Native object | A list of string hashmaps describing the requested services, or null value if there is an error. |

### getSituation

Fetches a specified Situation from the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|

| | |
|---|---|
| Object | A JavaScript object representing the Situation. |

### getSituationActions

Returns activity for specified Situations. Created by passing an object with the information requested. You can use the **from** and **to** arguments to specify a period that you want to retrieve Situation actions for. If you do not specify these, all actions are returned.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **sitn_ids** | JSON list | List of Situation IDs. |
| **start** | Number | Starting row from which data should be included. |
| **limit** | Number | Maximum number of actions you want to return. |
| **actions** | JSON list | List of action codes of actions you want to include in the return. If no action codes are specified, all action codes are returned. See Situation Action Codes for a list of action codes and their descriptions. |
| **from** | Number | Start time (in Unix epoch time) of the period you want to retrieve Situation actions for. |
| **to** | Number | End time (in Unix epoch time) of the period you want to retrieve Situation actions for. |

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript object containing the activity for specified situations |

Example

Input:

```
var actions = moogdb.getSituationActions(request);
```

Example **request** object to return the first 100 actions for Situation IDs 1, 2, and 3 for action codes 1 (Situation Created) and 14 (Added Alerts To Situation):

```
{
     "sitn_ids" : [1, 2, 3],
     "start" : 0,
         "limit" : 100,
         "actions" : [1, 14]
}
```

Successful return:

```
[{
    "uid": 2,
    "action_code": 1,
    "description": "Situation Created",
    "details": {},
    "type": "event",
    "sig_id": 1,
    "timed_at": 1507039842
    }, {
    "uid": 2,
    "action_code": 14,
    "description": "Added Alerts To Situation",
    "details": {}
    "alerts": [1, 2]
```

```
        }
    }]
```

### getSituationAlertIds

Returns the total number of alerts, and a list of their alert IDs for a specified Situation. This can be either all alerts or just those alerts unique to the Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **uniqueOnly** | Boolean | Gets alert IDs from the Situation:<br>true = get those alerts unique to the Situation<br>false = get all alerts in the Situation |

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript object containing the total and the alert IDs |

Example

Return:

```
{
    "total_alerts":10,
    "alert_ids":[4, 5, 6, 12, 14, 15, 16, 17, 18, 19]
}
```

### getSituationIds

Get all situation Ids matching the query.

Request Argument

| Name | Type | Description |
|---|---|---|
| **query** | JSON Object | A JSON Object containing the alert filter information |
| **limit** | Number | The maximum number of situation ids to return |

Return Parameter

| Type | Description |
|---|---|
| NativeObject | A Javascript object containing the total and the Situation IDs |

Example

Return:

```
{
    "total_situations":10,
    "sitn_ids":[4, 5, 6, 12, 14, 15, 16, 17, 18, 19]
}
```

### getSituationHosts

Returns a list of host names for a specified Situation, either for all the alerts in the Situation or just for the unique alerts.

Hosts are the names (defined in the **alerts.source** field in the database) for the sources of Events.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **uniqueOnly** | Boolean | Gets host names for the Situation:<br>true = get those host names unique to the Situation<br>false = all host names in the Situation |

Return Parameter

| Type | Description |
|---|---|
| **Native object** | A Javascript array containing the host names |

Example

Return:

```
{
    "hosts": [
        "server1",
        "server2",
        "server3",
        "server4",
        "server5",
        "server6",
        "server7"
    ]
}
```

### getSituationProcesses

Returns a list of process names for a specified Situation, and the primary process name, if defined.

Request Argument

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript array containing the process names, and the Situation's primary process, if defined |

Example

Return, with a primary process name defined:

```
{
    "processes": [
        "Process1",
        "Process2"
    ],
    "primary": "Process2"
}
```

### getSituationServices

Returns a list of external service names for a specified Situation, and the primary service name, if defined.

Request Argument

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript array containing the service names, and the Situation's primary service, if defined |

Example

Return, with a primary service name defined:

```
{
    "services": [
        "Service1",
        "Service2"
    ],
    "primary": "Service1"
}
```

### getSituationTopology

Returns the topology of all alerts connected to a Situation. This is sent as a JSON object in NetJSON format that represents the nodes affected by the Situation.

Request Argument

| Name | Type | Description |
|---|---|---|
| **sigId** | Number | The Situation ID |
| **contextLevel** | Integer | Level of contextual nodes to return |
| **topologyPropsObj** | Native array | Array of node properties to be returned. Valid properties are:<br>**severity**: Severity of the node.<br>**prc**: Whether this node is the probable root cause of the alert.<br>**service**: Service affected by the node.<br>**context**: Number of contextual hops between this node and a node directly affected by the Situation. A context of **0**means that the node is directly affected.<br>**description**: Description of the node.<br>**vertex_entropy**: Vertex Entropy of the node. |
| **fieldName** | String | Attribute of the alert that defines the node. The default is the alert 'source' but you can specify any valid alert field, including **custom_info** attributes. |

Return Parameter

| Type | Description |
|---|---|
| Object | A JSON object in NetJSON format that represents the nodes affected by the Situation. |

Example

Return, with a primary service name defined:

```
{
    "links": [
        {
            "source": "host2728",
            "target": "host2736"
        },
        {
            "source": "host2728",
            "target": "host1156"
        },
        {
            "source": "host2835",
            "target": "host2728"
```

```
            },
            {
                "source": "host2801",
                "target": "host2827"
            },
            {
                "source": "host2800",
                "target": "host2801"
            },
            {
                "source": "host2801",
                "target": "host2835"
            },
            {
                "source": "host2835",
                "target": "host2736"
            }
        ],
        "nodes": [
            {
                "id": "host2835",
                "properties": {
                    "severity": 5,
                    "prc": 0.9862626716344282,
                    "service": "",
                    "context": 0,
                    "description": "",
                    "vertex_entropy": 0.1794592472207979
                }
            },
            {
                "id": "host2736",
                "properties": {
                    "severity": 4,
                    "prc": 0.42722191049803876,
                    "service": "",
                    "context": 0,
                    "description": "",
                    "vertex_entropy": 0.08976540495989357
                }
            },
            {
                "id": "host2728",
                "properties": {
                    "severity": 3,
                    "prc": 0.007672752075071621,
                    "service": "",
                    "context": 0,
                    "description": "",
                    "vertex_entropy": 0.1794592472207979
                }
            },
            ]
    }
```


**getTeams**

A GET request that returns all teams created in the Cisco Crosswork Situation Manager instance.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |

Return Parameters

| Type | Description |
|---|---|
| Native Object | A native object containing information about all teams in Cisco Crosswork Situation Manager. |

Example

Curl Command:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getTeams"
```

Successful request return:


```
[
    {
        "room_id": 1,
        "alert_filter": "",
        "user_ids": [
            3
        ],
        "sig_filter": "",
        "landing_page": "",
        "description": "Example Team",
        "active": true,
        "team_id": 1,
        "services": [
            "Commerce",
            "Compute",
            "CRM",
            "Database",
            "Mobile",
            "Networking",
            "Remote",
            "Social",
            "Storage",
            "Switch",
            "Web"
        ],
        "users": [
            "admin"
        ],
        "name": "Cloud DevOps",
        "service_ids": [
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
            9,
            10,
            11
        ]
    },
    {
        "room_id": 2,
        "alert_filter": "",
        "user_ids": [
            3,
            5,
            7
```

```
            ],
            "sig_filter": "",
            "landing_page": "",
            "description": "",
            "active": true,
            "team_id": 2,
            "services": [
                "Compute",
                "Mobile",
                "Remote",
                "Storage",
                "Switch"
            ],
            "users": [
                "admin",
                "1",
                "3"
            ],
            "name": "DatabaseOps",
            "service_ids": [
                3,
                5,
                7,
                9,
                10
            ]
        }
    ]
```

### getTeam

A GET request that returns a team's details by team ID or name.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **team_id** | Integer | ID of the team to retrieve information about. |
| **name** | String | Name of a valid team to retrieve information about. |

Return Parameters

| Type | Description |
|---|---|
| JSON Object | A JSON Object containing details about the team. |

Examples

Request to return the details for the team with ID 1:

```
var teamData = moogdb.getTeam(1);
```

Successful request return:

```
{
    "room_id": 1,
    "alert_filter": "",
    "user_ids": [
        3
    ],
    "sig_filter": "",
    "landing_page": null,
    "description": "Example Team",
    "active": true,
```

```
    "team_id": 1,
    "services": [],
    "users": [
        "admin"
    ],
    "name": "Cloud DevOps",
    "service_ids": []
}
```

Example 2 (name)

Curl Command to return the details for team Cloud DevOps:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeam?name=Cloud DevOps"
```

Successful request return:

```
{
    "room_id": 1,
    "alert_filter": "",
    "user_ids": [
        3
    ],
    "sig_filter": "",
    "landing_page": null,
    "description": "Example Team",
    "active": true,
    "team_id": 1,
    "services": [],
    "users": [
        "admin"
    ],
    "name": "Cloud DevOps",
    "service_ids": []
}
```

### getTeamsForService

A GET request to return all teams related to the service with the specified ID or name.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid auth_token returned from the authenticate request. |
| **service_id** | String | The ID of the service. |
| **name** | String | The name of the service. |

Return Parameters

| Type | Description |
|---|---|
| Native Object | A native object containing information about all teams in associated with the specified services in Cisco Crosswork Situation Manager. |

Examples

Curl Command for service_id:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamsForService?service_id=1"
```

Curl command for service name:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getTeamsForService?service_name=web"
```

Successful request return:


```
[
    {
        "room_id": 1,
        "alert_filter": "",
        "user_ids": [
            3
        ],
        "sig_filter": "",
        "name": "Cloud DevOps",
        "landing_page": "",
        "description": "Example Team",
        "active": true,
        "service_ids": [
            1,
            2,
            3,
            4,
            5,
            6,
            7,
            8,
            9,
            10,
            11
        ],
        "team_id": 1,
        "services": [
            "Commerce",
            "Compute",
            "CRM",
            "Database",
            "Mobile",
            "Networking",
            "Remote",
            "Social",
            "Storage",
            "Switch",
            "Web"
        ],
        "users": [
            "admin"
        ]
    }
]
```


### getTeamSituationIds

Get all situation Ids for the given team.

Request Argument

| Name | Type | Description |
|---|---|---|
| **teamName** | String | The team name |
| limit | Number | The number of situations to return |

Return Parameter

| Type | Description |
|---|---|
| NativeObject | A Javascript object containing the total and the Situation IDs |

Example

Return:

```
{
     "total_situations":10,
     "sitn_ids":[4, 5, 6, 12, 14, 15, 16, 17, 18, 19]
}
```

### getThreadEntries

Returns thread entries for the specified Situation. Threads are comments or 'story activity' on Situations.

You can select specific thread entries to return using start and limit values. If not, the first 100 entries will be returned. The entries returned are ordered by most recent entries first.

Request Arguments

| Name | Type | Required |
|---|---|---|
| **situationId** | Number | Required |
| **thread** | String | Required |
| **start** | Number | Optional |
| **limit** | Number | Optional |

Return Parameter

| Type | Description |
|---|---|
| Native object | A Javascript object containing details of the selected thread entries |

Examples

Request to get the thread entries for thread "Support" on Situation ID 58:

```
var threadEntries = moogdb.getThreadEntries(58, “Support”, 0, 10);
```

Successful return:

```
{
  "entries": [
               {
                   "uid": 3,
                   "entry": "This one is important. Another comment",
                   "agrees": [],
                   "total_comments": 0,
                   "thread_id": "Support",
                   "mmid": -1,
                   "sig_id": 58,
                   "entry_id": 2,
                   "timed_at": 1423226829,
                   "disagrees": [],
                   "commenters": []
               },
               {
                   "uid": 3,
                   "entry": "No comment. A comment",
                   "agrees": [],
                   "total_comments": 0,
                   "thread_id": "Support",
                   "mmid": -1,
                   "sig_id": 58,
                   "entry_id": 1,
                   "timed_at": 1423226807,
                   "disagrees": [3],
                   "commenters": []
               }
```

```
            ],
  "total_entries": 2
}
```

### getUser

Fetches user information from the database, given the user ID or username.

Request Argument

| Name | Type | Description |
|---|---|---|
| **userId** | Number | A valid user ID |
| **username** | String | A valid username |

Return Parameter

| Type | Description |
|---|---|
| CEvent | A CEvent object containing the user information |

Example:

Example request:

```
var cevent = moogdb.getUser(6);
```

Example response:

```
{active=true, competencies=[], contact_num=, department=null,
description=Online, email=, fullname=cyber, groupname=End-User,
invitations=[], joined=1516963803, only_ldap=0, photo=-1, primary_group=1,
profile_image=null, realms=[DB], roles=[1, 3, 4, 5], session_expiry=null,
status=1, teams=[], timezone=SYSTEM, uid=6, username=cyber}
```

### getUsers

Fetches all users from the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| **limit** | Integer | The number of users to return. 1000 by default. |

Return Parameter

| Type | Description |
|---|---|
| NativeObject | A JavaScript list of objects describing the users. |

Example

Return:

```
[
    {
        "uid": 3,
        "teams": [
            "Cloud DevOps"
        ],
        "fullname": "Administrator",
        "username": "admin"
    },
    {
        "uid": 6,
        "teams": [],
        "fullname": "Nagios",
        "username": "Nagios"
```

```
        },
        {
            "uid": 5,
            "teams": [],
            "fullname": "Webhook",
            "username": "Webhook"
        }
    ]
```

**getUserName**

Fetches user information from the database, given the user ID.

Request Argument

| Name | Type | Description |
|---|---|---|
| **userId** | Number | A valid user ID |

Return Parameter

| Type | Description |
|---|---|
| String | The corresponding username for the submitted user ID. |

getUserRoles

Fetches the user's roles from the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| userid | Number | A valid userId |
| username | String | A valid username |

Return parameter

| Type | Description |
|---|---|
| NativeObject | A JavaScript object containing Role id, Role name and Role description |

Example

Return:

```
[{
        "id": 1,
        "name": "Super User",
        "description": "Super User"
}, {
        "id": 3,
        "name": "Manager",
        "description": "Manager"
}, {
        "id": 4,
        "name": "Operator",
        "description": "Operator"
}]
```

**getUserTeams**

Fetches the user IDs and team names for a specified user in the database.

Request Argument

| Name | Type | Description |
|---|---|---|
| userid | Number | A valid user ID. |

| | | |
|---|---|---|
| username | String | A valid username |

Return parameter

| Type | Description |
|---|---|
| CEvent | A CEvent containing the team IDs and team names. |

```
[{
        "id": 2,
        "name": "Alpha"
}, {
        "id": 3,
        "name": "Epsilon"
}, {
        "id": 4,
        "name": "Moo_team"
}]
```

### getWorkflowEngineMoolets

Request Arguments

None.

Return Parameter

A JSON array of moolet objects. Each object has the following:

| Name | Type | Description |
|---|---|---|
| moolet_name | String | The Moolet name. |
| moolet_type | ENUM/String | The Moolet type: event, alert, or situation. |
| active | Boolean | Is the workflow engine that the Moolet represents active? |
| functions[c][d][e] | JSON | The available functions in the moobot - each key is the function name and the values are:<br><br>Description (String) -- The description of the function<br><br>Decision (Boolean) -- If True, treat the result of this function as a decision,.<br><br>Arguments - (JSON) the arguments of the function, a map from the argument name to:<br><br>Type - (ENUM/String) the type of argument - either Text, JSON or Number.<br><br>Description - Human readable description of the argument. |
| last_updated | Integer | UNIX time when the Moolet was last updated. |

Example

```
[{
    "moolet_name": "Alerts Workflows",
    "moolet_type": "alert",
    "active": true,
    "functions": {
            "functionOne": {
                    "description": "The first function",
                    "decision": true,
                    "arguments": {
                            "severity": {
                                    "type": "Number",
                                    "description": "The severity."
```

```
                        }
                    }
                },
                "functionTwo": {
                    "description": "The second function",
                    "decision": false,
                    "arguments": {
                        "customInfo": {
                            "type": "JSON",
                            "description": "The custom info."
                        },
                        "key": {
                            "type": "Text",
                            "description": "The key within the
custom info."
                        }
                    }
                }
        },


        "last_updated": 1545306590
}]
```

### getWorkflows

Get all the known workflows by moolet name.

Request Arguments

| Name | Type | Description |
|---|---|---|
| mooletName | String | Required<br><br>Name of the moolet to retrieve workflows for. |
| activeOnly | Boolean | Return only the active workflows. |

Return Parameter

JSON array of matching workflows, where each has:

| Type | Type |
|---|---|
| id | Integer |
| moolet_name | String |
| workflow_name | String |
| sequence | Integer |
| active | Boolean |
| description | String |
| entry_filter | JSON filter |
| sweep_up_filter | JSON filter |
| first_match_only | Boolean |
| operations | JSON list |

Example

```
[{
        "id": 1,
```

```
        "moolet_name": "Alerts Workflows",
        "workflow_name": "ChangeInfoWorkflow",
        "sequence": 1,
        "active": true,
        "description": "Changingthealertinformation",
        "entry_filter": {
                "column": "severity",
                "op": 5,
                "value": 3,
                "type": "LEAF"
        },
        "sweep_up_filter": {
                "column": "description",
                "op": 4,
                "value": "description",
                "type": "LEAF"
        },
        "first_match_only": true,
        "operations": [{
                        "type": "action",
                        "function_name": "functionA",
                "operation_name": "Name of operation",
                        "function_args": {
                                "admin": 2
                        }
                },
                {
                        "type": "delay",
                        "delay": 30,
                "reset": false
                }
        ]
    }]
```

### mergeSituations

Merges two or more Situations, superseding the originals if required, and returning the newly created Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationIds** | Native array | A Javascript array containing the IDs of the Situations to merge |
| **keepOriginals** | Boolean | Determines what to do with the original Situations:<br><br>true = keep the original Situations<br><br>false = supersede the original Situations |

Return Parameter

| Type | Description |
|---|---|
| CEvent | A CEvent object containing the newly created Situation |

### moveSituationToCategory

Move a Situation into a new category.

A category represents a type of Situation, indicating how it was created or its state. See ‘Create Shared Alert and Situation Filters’ in *Cisco Crosswork Situation Manager Administrator Guide* for more information.Create Shared Alert and Situation Filters

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **category** | String | The name of the new category |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### moveSituationToQueue

Assigns a specified Situation to a queue and writes a thread entry if required. The queue and user may be provided as either an ID or a valid name.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **user** | Object | An object containing either a valid user name or ID |
| **queue** | Object | An object containing either a valid queue name or ID |
| **journal** | String | An entry to add to the journal thread, if required<br><br>Optional |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### removeAlertFromSituation

Removes a specified Alert from a Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | The Alert ID |
| **situationId** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### removeSigCorrelationInfo

Removes all correlation information related to a specified Situation.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **auth_token** | String | A valid **auth_token** returned from the **authenticate** request. |
| **sitn_id** | Number | The Situation ID |
| **serviceName** | String | The service name (Optional). |
| **externalId** | String | The external ID (Optional). |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### reorderWorkflows

Reorder the sequence of Workflows within a Moolet.

Request Arguments

| Argument | Type | Description |
|---|---|---|
| moolet_name | String | Required -- Moolet name. |
| workflow_IDs_sequence | Array of Integers | Required -- An ordered array of all the workflow IDs, where position 0 is the first ID in the sequence. |

Return Parameters

| Type | Description |
|---|---|
| Boolean | True if the operation was successful. |

Example

```
moogdb.reorderWorkflows("Alerts Workflows", [1, 4, 3, 2, 5]);
```

### resolveSituation

Resolve a specified Situation that is currently open.

Request Argument

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### reviveSituation

Revive (set to open) a specified Situation that is currently set to resolved.

Request Argument

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### setAlertCustomInfo

Updates the custom information in the database for specified Alert.

This method can either be used with the **alertInfo** CEvent or with both the **alertID** and **customInfoMap** arguments.

The **merge** parameter can be used alongside either methods. This determines whether to merge the new custom information data with existing data or replace it.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | The Alert ID.<br>Note:<br>Can be used alongside **customInfoMap** and **merge** but not **alertInfo**. |
| **alertInfo** | CEvent | A CEvent containing **alert_id** and **custom_info** attributes, the values of which will be used to replace the **custom_info** in the specified Alert.<br>Note:<br>Can be used alongside **merge** but not **alertId** or **customInfoMap**. |
| **customInfoMap** | Object | A map of name value pairs containing the new **custom_info** information. |
| **merge** | Boolean | Determines what is done with the custom information:<br>true = merge the existing data with the new data<br>false = replace the existing data with the new data. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail. |

### setAlertSeverity

Sets the severity level for a specified Alert.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | The Alert ID |
| **severity** | Number | The Alert's severity as an integer:<br>0 Clear<br>1 Indeterminate<br>2 Warning<br>3 Minor<br>4 Major<br>5 Critical |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail. |

### setPrcLabels

Updates the probable root cause (PRC) information for specified alerts within a Situation. You must specify at least one alert ID and a PRC level for the alert.

You can mark alerts as causal, non_causal or unlabelled within a Situation. An alert can have different PRC levels within different Situations.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **alert_ids** | JSON list | A list of the alert IDs |
| **causal**<br>**non_causal**<br>**unlabelled** | JSON list | PRC levels |

Input example:

```
var prcLabels = { causal: [1], unlabelled: [4], non_causal: [2,3] };
moogdb.setPrcLabels(1, prcLabels);
```

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### setResolvingThreadEntry

Sets or clears a thread entry in a Situation as a resolving step. Threads are comments or 'story activity' on Situations.

This method returns a Boolean indicating whether the thread entry was successfully set or cleared as a resolving step.

Request Arguments

| Name | Type | Required |
|---|---|---|
| **entryId** | Number | Yes |
| **resolving_step** | Boolean | Yes |
| **userId** | Number | Yes |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Whether or not the thread entry was successfully set or cleared as a resolving step. |

Examples

Request to set thread entry 32 as a resolving step using user ID 1:

```
var success = moogdb.setResolvingThreadEntry(32, true, 1);
```

Return of successful request:

```
true
```

### setSigCustomInfo

Updates the custom information in the database for specified Situation.

The Situation ID and new custom information are both contained in the **situationInfo** CEvent.

The new custom information is contained in the **customInfoMap** object.

The **merge** parameter determines whether to merge the new custom information data with existing data or replace it.

Request Arguments

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **situationId** | Number | The Situation ID |
| **customInfoMap** | Object | A map of name value pairs containing the new **custom_info** information. |
| **merge** | Boolean | Determines what is done with the custom information:<br>true = merge the existing data with the new data<br>false = replace the existing data with the new data |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### setSituationProcesses

Applies a list of processes (contained in the **processes** Javascript array) to a specified Situation.

Any other processes already associated with the Situation are removed.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID. |
| **processes** | Native array | A Javascript array containing the process names. If any processes supplied do not exist in the database, they are created and assigned to the Situation. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### setSituationServices

Applies a list of external services (contained in the services JavaScript array) to a specified Situation.

Any other services already associated with the Situation are removed.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **situationId** | Number | The Situation ID. |
| **services** | Native array | A JavaScript array containing the service names. If any services supplied do not exist in the database, they are created and assigned to the Situation. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### updateAlert

Takes an Alert object and uses it to update the database and the MooMS bus.

Request Argument

| Name | Type | Description |
|---|---|---|
| **alertObject** | CEvent | The Alert object |

Return Parameter

| Type | Description |
|---|---|

| | |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### updateCustomInfo

Update the custom info for an alert or Situation.

Request Argument

| Name | Type | Description |
|---|---|---|
| **toUpdate** | CEvent | A CEvent representing the alert or Situation you want to update. |
| **toMerge** | JavaScript Object | The custom info to add to/replace the existing custom info field. |
| **merge** | Boolean | Merge the existing and new custom info if true. Replaces existing custom info if false. Defaults to true. |

For an alert you can also use the following arguments:

| Name | Type | Description |
|---|---|---|
| **alertId** | Number | Alert ID of the alert you want to add custom info to. |
| **path** | String | Dot-notation path to the **custom_info** key where the info is stored. Updates existing value if the key already exists; creates the full path if the key does not exist. |
| **param** | Value | Value to put at the specified key. |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful. True = success, false = fail. |

### updateMaintenanceWindow

Updates a maintenance window object, by passing an object containing the maintenance window information.

Request Argument

| Name | Type | Required | Description |
|---|---|---|---|
| updatedWindow | Native object | Yes | Maintenance window object containing the updated details. |

The maintenance window object **maintenanceWindowObj** contains the following information:

| Name | Type | Required | Description |
|---|---|---|---|
| window_id | Number | Yes | ID of the maintenance window. |
| name | String | Yes | Name of the maintenance window. |
| description | String | Yes | Description of the maintenance window. |
| filter | String | Yes | Filter to apply to the new alerts created. |
| start_date_time | Number | Yes | The time in epoch when the maintenance window will start, up to a maximum of 5 years in the future. |
| duration | Number | Yes | Duration, in seconds, that the maintenance window will run for. Must be greater than zero. |
| forward_alerts | Boolean | Yes | Whether or not alerts will be forwarded to a Situation. |
| recurring_period | Number | No | Whether or not this is a recurring maintenance window. Set this to:<br>**1** for a recurring maintenance window. |

| | | | |
|---|---|---|---|
| | | | **0** for a one-time maintenance window.<br><br>If you change this from **0** to **1**, you must specify **recurring_period_units**. |
| recurring_period_units | Number | No | Specifies the recurring period of the maintenance window, in days, weeks or months. If you set **recurring_period** to **0**, you must set **recurring_period_units** to **0**. Valid values are:<br><br>**0** = a one-time maintenance window<br><br>**2** = daily<br><br>**3** = weekly<br><br>**4** = monthly |

Example

Request to update a maintenance window:

```
var updatedWindow = moogdb.updateMaintenanceWindow(windowToUpdate)
```

Where **windowToUpdate** is as follows:

```
{
  "window_id":351,
  "name":"Updated name",
  "description":"Updated Description",
  "filter":"source = \"server1\"",
  "start_date_time":1546433400,
  "duration":3600,
  "forward_alerts":false,
  "recurring_period":1,
  "recurring_period_units":3
}
```

### updateSituation

Takes a Situation object and uses it to update the database and the Message bus.

Request Argument

| Name | Type | Description |
|---|---|---|
| **situationObject** | CEvent | The Situation object |

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

### updateTeam

Update the team, by passing an object containing team information.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **teamObj** | Object | A map containing the following team information |
| **team_id** | Number | Mandatory - The team ID |
| **name** | String | Optional - The new team name. Leave empty to leave Cisco Crosswork Situation Manager as is |

| | | |
|---|---|---|
| **alert_filter** | String | Optional - The new team alerts filter. Either a SQL like filter or an JSON representation of the filter. Leave empty to leave Cisco Crosswork Situation Manager as is |
| **services** | JSON List | Optional - List of the team services names or IDs. Leave empty to leave Moogsoft AIOps as is |
| **sig_filter** | String | Optional - The situation filters. Either a SQL like filter or an JSON representation of the filter. Leave empty to leave Moogsoft AIOps as is |
| **landing_page** | String | Optional - The team default landing page. Leave empty to leave Moogsoft AIOps as is |
| **active** | Boolean | Optional - False if the team is inactive, true if the team is active. Default to true. Leave empty to leave Moogsoft AIOps as is |
| **description** | String | Optional - The team description. Leave empty to leave Moogsoft AIOps as is |
| **users** | List of numbers or strings | Optional - The team users (either IDs or usernames). Leave empty to leave Moogsoft AIOps as is |

Input example

```
{
    "team_id" : 3,
        "name": "myTeam",
        "alert_filter": "{ \"column\": \"count\", \"op\": 1, \"value\": 1,
\"type\": \"LEAF\" }",
        "sig_filter": "{ \"column\": \"severity\", \"op\": 1, \"value\":
5, \"type\": \"LEAF\" }",
        "active": true,
        "services": [1, 2, 4],
        "users": ["user1", "user4"],
        "description": "myDescription",
        "landing_page": ""
}
```

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

**updateUser**

Update the user, by passing an object containing user information.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **userObj** | Object | A map containing the following user information |
| **username** | String | Mandatory (optional if user id used) - the user login username |
| **uid** | Number | Mandatory (optional if username used) - the user id |
| **password** | String | The new user password (only valid for DB realm) |
| **active** | Boolean | true if the user active, false if the user inactive, default to true |
| **email** | String | The user email address |
| **fullname** | String | The user full name |

| | | |
|---|---|---|
| **roles** | JSON list | List of user roles. That list should contain either the list the role IDs or the role names. For example, "roles":["Super User"], |
| **primary_group** | String or Number | The user primary group name or primary group id |
| **department** | String or number | The user department id or name |
| **timezone** | String | The user timezone |
| **contact_num** | String | The user phone number |
| **session_expiry** | Number | The number of minutes after which the user session will expire. Default to system default |
| **competencies** | JSON list | A list with the user competencies. Each competency should have the name or cid and ranking. That is, something like:<br>**[**<br>**{ "name" : "SunOS" , "ranking" : 40 },**<br>**{ "name" : "SAP" , "ranking" : 50 },**<br>**{ "name" : "EMC" , "ranking" : 60 }**<br>**]** |
| **teams** | JSON list of numbers or strings | List of the user teams. The list should contains either the list of the teams ID or the teams name. |

Input example :

```
{
        "uid": 5,
        "fullname": "firstName surName",
        "competencies": [{
                        "name": "SunOS",
                        "ranking": 40
                },
                {
                        "name": "SAP",
                        "ranking": 50
                },
                {
                        "name": "EMC",
                        "ranking": 60
                }
        ],
        "roles": ["Super User"],
        "department": 3,
        "active": true,
        "email": "user@email.com",
        "timezone": "a timezone",
        "teams": [1, 2, 4],
        "joined": 12345678,
        "contact_num": "0965412345"
}
```

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the operation was successful: true = success, false = fail |

**updateWorkflows**

Update one or more existing workflows.

Request Arguments

| Name | Type | Description |
|---|---|---|
| id | integer | The ID of the workflow to be updated. |
| details | JSON | A details object with the following fields. |

Details Object

| Name | Type | Description |
|---|---|---|
| workflow_name | String | Workflow name. |
| active | Boolean | If true, the workflow is active. |
| description | String | Workflow description. |
| entry_filter | Filter in JSON or SQL format | The sweep-up filter. An empty, null, or missing filter means no sweep-up. |
| first_match_only | Boolean | Perform workflow operations only once on each object. |
| sweep_up_filter | Filter in JSON or SQL format | The sweep-up filter. An empty, null, or missing filter means no sweep-up. |
| operation | JSON list | A list of operations, each being:<br><br>type - (ENUM/String) (Mandatory) the type of the operation. Either action, decision, delay. Depending on the type, we'll read other fields in the JSON<br><br>operation_name - (String, Required for action/decision type) The operation name.<br><br>* function_name - (String, Required for action/decision type) The function name.<br><br>* function_args - (JS Object) (*Optional for action/decision type*) The arguments for the action or decision function.<br><br>duration - (Integer) (Required for delay type) The amount of seconds before the message goes to the next operation/Workflow/moolet |
| reset | Boolean | Mandatory for delay type |

Return Parameter

| Type | Description |
|---|---|
| Boolean | True if the operation was successful. |

Request Example

```
moogdb.updateWorkflows(1, {workflow_name: "new name"});
```

## Moolet Informs

- Before You Begin
- Configure the Module
- Reference
  - create
  - setSubject

- setPayload
- setDetails
- setTarget
- send

- Example

  You can configure a Moolet to exchange messages about update information with other Moolets using the Moolet Informs module. For example, after you label some alerts you can configure the module to inform the ticketing Moolet to update the severity of a ticket based on the new label.

  To enable this functionality, you add the Moolet Informs module at the start of a Moobot associated with the Moolet you want to send inform messages from.

### Before You Begin

Before you get started, ensure you have met the following requirements:

- You have a Moolet and associated Moobot you want to send inform messages from.
- You know the Moolets you want to receive the inform messages. These are your "targets".

### Configure the Module

To use the Moolet Informs module:

1. Go to the Moobot associated with the source Moolet from which you want to send Inform messages. Load the module at the top of the file:

   ```
   var mooletInforms = MooBot.loadModule('MooletInforms');
   ```

2. Create the Moolet Inform using the create method as follows, passing the target Moolets that receive messages from this source:

   ```
   var inform = mooletInforms.create("AlertRulesEngine", "Cookbook");
   ```

   Specifying the target Moolets is not required in this step. However, you will need to specify the targets later.

   ```
   var inform = mooletInforms.create();
   ```

3. Add values to the inform using one or more of the following:

   - **inform.setSubject**: Subject of the inform message. You can use this to enable a different workflow within the target Moobot.
   - **inform.setPayload**: Any CEvent object. See Events for more information.
   - **inform.setDetails**: Details of any other data you want to send as a JSON object.

4. If you did not specify the target Moolets previously, specify them now:

   - **inform.setTarget**: List of Moolets the messages are sent to by the Moolet.

5. There are two ways to configure how the messages are sent. If you have already set your targets:

   ```
   inform.send();
   ```

   If you have not set your targets, include them in the method call:

```
inform.send("AlertRulesEngine", "Cookbook");
```

6. Go to the config file for each target Moolet and add an event handler to listen for the Inform messages:

```
events.onEvent("informReceive",
constants.eventType("mooletInforms.ExampleMoolet")).listen();
```

7. There are two ways for the listening target Moolet to access the data.

```
function informReceive(inform) {
    var subject = inform.getSubject();
    var payload = inform.getPayload();
    var details = inform.getDetails();
        logger.warning("Received Moolet Inform. Subject [" + subject + "]
Payload [" + payload + "] Details [" + details + "]");
}
```

Alternatively, you can use the value method:

```
function informReceive(inform) {
    var subject = inform.value("subject");
    var payload = inform.value("payload");
    var details = inform.value("details");
    logger.warning("Received Moolet Inform. Subject [" + subject + "]
Payload [" + payload + "] Details [" + details + "]");
}
```

8. You can configure the Moolet to call a specific method for different subjects in the inform messages. For example you can configure a Remedy Moolet to listen for a specific subject in the inform message and route the event to a function:

```
events.onEvent("createNewTicket",
constants.eventType("mooletInforms.RemedyMoolet.ticketCreate")).listen();
```

After you have completed your configuration, inform messages are sent to your target Moolets which will call any methods you have added.

## Reference

You can use the following methods in the Moolet Informs module:

**create();**

Creates the Moolet inform message. You can choose to select one or more Moolet targets to receive the messages or you can leave this empty.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **targets** | String | No | A single or comma separated list of Moolet names to target. |

Return Parameter

A new Moolet Inform Java object

Example

```
var inform = mooletInforms.create("MaintenanceWindowManager",
"AlertRulesEngine");
```

### setSubject

Set the name of the topic for the Moolets to listen for on the Message Bus.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **Subject** | String | Yes | Name of subject the Moolets listen for on the Message Bus. |

Return Parameter

None

Example

```
inform.setSubject("subTopic");
```

### setPayload

Any CEvent object . See Events for more information.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| Payload | CEvent | Yes | Any CEvent object that has been passed into the Moobot from the pipeline, or has been retrieved from MoogDb. |

Return Parameter

None

Example

```
inform.setPayload(event);
```

### setDetails

Details of any other data you want to send in the Moolet inform message.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| setDetails | NativeObject | Yes | A JSON object containing any details you want to send. |

Return Parameter

None

Example

```
inform.setDetails({"signature":"Loss of Signal","description":"Loss of Signal","source":"S-DF_P2_1"});
```

### setTarget

Set the target Moolets you want to receive the Moolet inform messages. Use this method if you did not set the target Moolets with the create method.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **targets** | String | Yes | A single or comma separated list of Moolet names to target. |

Return Parameter

None

Example

```
inform.setTarget("AlertRulesEngine", "Cookbook");
```

#### send

Sends the Moolet inform messages to your target Moolets.

Request Arguments

| Name | Type | Required | Description |
|---|---|---|---|
| **targets** | String | No | The name or names of Moolets to target this message to. |

Return Parameter

None

Example

```
inform.send("Cookbook");
```

### Example

An example of a Moolet Inform that sends a signature, description and source to the Cookbook Moolet:

```
var mooletInforms = MooBot.loadModule('MooletInforms');
var inform = mooletInforms.create();
inform.setSubject("subTopic");
inform.setPayload(event);
inform.setDetails({"signature":"Loss of Signal","description":"Loss of
Signal","source":"S-DF_P2_1"});
inform.send("Cookbook");
}
```

An example of how to configure the listener or target Moolet:

```
events.onEvent("handleEvent",
constants.eventType("mooletInforms.EmptyMoolet.event_subject")).listen();
events.onEvent("handleAlert",
constants.eventType("mooletInforms.EmptyMoolet.alert_subject")).listen();
events.onEvent("handleSig",
constants.eventType("mooletInforms.EmptyMoolet.sig_subject")).listen();
```

## Moolet Information API

You can use the following commands in a Moobot file to obtain contextual information about the associated Moolet. These commands are useful in automation and other workflows where you want to verify the Moolet context before performing an action such assending data or some other action.

#### Bot.getType()

Return the Moolet type. If the result is Bot.WORKFLOW_ENGINE, you can call Bot.WorkflowEngine.getMessageType() to find the workflow-engine type.

Request Arguments

None.

Return Parameter

| Type | Description |
|---|---|
| Enumerated type | Can be one of the following:<br><br>Bot.ALERT_BUILDER<br><br>Bot.ALERT_RULE_ENGINE |

| | |
|---|---|
| | Bot.COOKBOOK<br>Bot.EMPTY_MOOLET<br>Bot.NOTIFIER<br>Bot.SCHEDULER<br>Bot.SITUATION_MANAGER<br>Bot.TEAMS_MANAGER<br>Bot.WORKFLOW_ENGINE |

Example

```
var MooletType = Bot.getType();
logger.warning(' Moolet type is ...' +MooletType);
```

**Bot.getMooletName()**

Return the Moolet type

Request Arguments

None.

Return Parameter

| Type | Description |
|---|---|
| String | Name of the associated Moolet. |

Example

```
if((Bot.GetType()MooletType === Bot.EMPTY_MOOLET ))
   {logger.warning(Bot.getMooletName()
                   + ' is an empty moolet')};
```

**Bot.WorkflowEngine.getMessageType()**

Return the workflow engine type, or null for non-workflow-engine Moolets.

Request Arguments

None.

Return Parameters

| Type | Description |
|---|---|
| String | Can be one of the following:<br>Bot.WorkflowEngine.ALERT<br>Bot.WorkflowEngine.SITUATION<br>Bot.WorkflowEngine.EVENT<br>null (if the associated Moolet is not a workflow engine) |

Example

```
if((Bot.getType() === Bot.WORKFLOW_ENGINE) &&
(Bot.workflowEngine.getMessageType() === Bot.workflowEngine.ALERT))
    {logger.warning('Moolet ' + Bot.getMooletName() + ' will handle
alerts')}
```

# Process

## Description

The Process module allows you to run and control the execution of another process.

The Process module is available to load into any standard Moobot.

To use, at the top of a Moobot js file, define a new global object **proc** to load the Process module:

```
var proc = MooBot.loadModule('Process');
```

Create a new process with **create** and access methods to run the process with **arg**

Then run the process in one of two ways - either **run** in a separate child process of moog_farmd, or **runToExit** run and only return when the process exits.

Stop processes running with **terminate.**

These methods are detailed below.

## Reference Guide

proc.create()

Defines a valid pathname to an executable file that you have permission to execute (or the user that started Moogfarmd has permissions to execute).

Request Argument

| Name | Type | Description |
|---|---|---|
| **process** | String | A pathname to an executable file (with permission). |

Return Parameter

| Name | Type | Description |
|---|---|---|
| **processObj** | Object | An object containing the process to run. |

proc.arg()

Access a series of methods by passing strings representing command line arguments required to run the process.

Request Arguments

| Name | Type | Description |
|---|---|---|
| **argString** | Strings | A list of strings representing command line arguments required to run the process. |

Return Parameter

Void - no value returned

proc.run()

Takes the object returned from **create** and runs the process in a separate child process of moog_farmd.

Request Argument

| Name | Type | Description |
|---|---|---|
| **processObj** | Object | The object returned from the **create** method. |

Return Parameter

| Type | Description |
|---|---|
| Object | An object containing the process results. |

proc.runToExit()

Takes the object returned from **create**, runs the process and only returns when the process exits.

Request Argument

| Name | Type | Description |
|---|---|---|
| **processObj** | Object | The object returned from the **create** method. |

Return Parameter

| Type | Description |
|---|---|
| Object | An object containing the process results. |

proc.terminate()

Stops the created processes running (causes the process under the process object returned from **create** to be terminated)

Request Argument

| Name | Type | Description |
|---|---|---|
| **processObj** | Object | The object returned from the **create** method |

Return Parameter

Void - no value returned.

Example

The following function runs an external tool **toolName** using the Process module:

```
function runTool(toolname,toolArgs,toExit)
{
var toolRun=proc.create(toolName);
        for ( var argIdx = 0; argIdx < toolArgs.length ; argIdx++)
        {
                toolRun.arg(toolArgs[argIdx]);
        }
        if ( toExit === true )
        {
                proc.runToExit(toolRun);
                var toolResults=toolRun.output();
                toolResults=toolResults.replace("\n","");
                return(toolResults);
        }
        else
        {
                proc.run(toolRun);
                return;
        }
}
```

Usage:

```
var toolScript = "/usr/share/moogsoft/scripts/hip_chat.py";
var toolArgs = [ "--room=","Support Team", "--sigid=",sigId ];
var hipChatData = runTool( toolScript,toolArgs, true );
```

This calls the tool runner, gets data back, runs the process as 'run to exit' (runToExit = true).

## RabbitMQ

- Configure the Module
- Reference Guide
    - connect

- send
- close

- Examples

  The RabbitMQ module allows you to broadcast information on a RabbitMQ bus. For example, you can use it to push alert or Situation data to a data warehouse via RabbitMQ.

  You cannot connect the RabbitMQ Moobot module to the RabbitMQ instance used by Cisco Crosswork Situation Manager.

### Configure the Module

To use the RabbitMQ Moobot module:

1. Define a new global object **`rabbit`** at the top of a Moobot JavaScript file to load the module.
2. Use the **`connect`** method to create a new connection to one or more RabbitMQ brokers.
3. Use the **`send`** method to send the required information.
4. Use the **`close`** method to close the connection.

Refer to the Examples for more details.

### Reference Guide

You can use the following methods in the RabbitMQ Moobot module.

connect

Establishes a connection to one or more RabbitMQ brokers with defined connection properties.

You cannot connect the RabbitMQ Moobot module to the RabbitMQ instance used by Cisco Crosswork Situation Manager.

Request Argument

| Name | Type | Description |
|---|---|---|
| **`<properties>`** | Object | A JavaScript object containing connection properties. See below. |

RabbitMQ Connection Properties

The RabbitMQ module **`connect`** method defines connection properties as a Javascript object, which may include the following keys:

| Key | Description |
|---|---|
| **`brokers`** | Top-level container for one or more target RabbitMQ brokers. For each broker, define:<br>**`host`**: Hostname or IP address of the RabbitMQ broker.<br>**`port`**: Port of the RabbitMQ broker. |
| **`user`** | Username to connect to RabbitMQ. |
| **`password`** | Username to connect to RabbitMQ. |
| **`timeout`** | Length of time to wait before halting a connection or read attempt, in milliseconds. Defaults to 10,000. |
| **`vhost`** | Name of the RabbitMQ virtual host. Optional. |
| **`ssl`** | Top-level container for the SSL configuration. Optional. |

| | |
|---|---|
| **ssl_protocol** | The SSL protocol to use. If not specified, TLSv1.2 is used by default. |
| **server_cert_file** | Name of the SSL root CA file. |
| **client_cert_file** | Name of the SSL client certificate. |
| **client_key_file** | Name of the SSL client key file. Must be in PKCS#8 format.<br><br>Note : Refer to Message System SSL in *Cisco Crosswork Implementor Guide* for more information.Message System SSL. |

Return Parameter

| Type | Description |
|---|---|
| Object | A Java object containing connection details, depending on the requested connection properties.<br><br>Returns **null** if no connection can be made. |

Example

```
{
    brokers: [
                {
                        host: "rabbithost",
            port: 5672
        }
    ],
    user: "rabbitmq_admin",
    password: "78smr9!b",
    timeout: 10000,
    vhost: "rabbitvhost",
    ssl: {
                ssl_protocol: "TLSv1.2",
        server_cert_file: "server.pem",
        client_cert_file: "client.pem",
        client_key_file: "client.key"
    }
}
```

send

Sends a message to the RabbitMQ broker. Refer to the basic class in the RabbitMQ AMQP 0-9-1 Reference for a list of keys that you can specify in the message properties.

| Name | Type | Description |
|---|---|---|
| **Exchange** | String | The RabbitMQ exchange. |
| **RoutingKey** | String | The RabbitMQ routing key. |
| **Properties** | String or Object | Message properties in one of the following formats:<br><br>Plain text<br><br>JSON Object payload<br><br>JSON Array payload |
| **Message** | String | The message to send. |

Return Parameter

None.

Examples

```
connection.send("direct_logs", "severity",
        {
```

```
                content-type : "text/xml",
                reply-to         : "greetings.hi",
                headers          : {"server" "app5.myapp.megacorp.com"
"cached" false}
        },
        "<Priority>1</Priority>"
)

connection.send("topic_logs", "topic", {contentType: "text/xml"},
"<Priority>1</Priority>");
```

close

Closes the connection to the RabbitMQ broker.

Request Argument

None.

Return Parameter

| Type | Description |
|---|---|
| Boolean | Indicates if the close operation was successful: true = success, false = fail. |

## Examples

The following examples demonstrate the use of the RabbitMQ Moobot modules:

```
var rabbit = MooBot.loadModule('RabbitMQ');
var connection = rabbit.connect({
   brokers:[
      {
         host:"myHost",
         port:5672
      }
   ],
   user:"test",
   password:"test",
   timeout:10000,
   vhost:"myVHost",
   ssl:{
      ssl_protocol:"TLSv1.2",
      server_cert_file:"server.pem",
      client_cert_file:"client.pem",
      client_key_file:"client.key"
   }
});


if (connection) {
    connection.send("test", "test", {contentType: "text/xml"},
"<testKey>testValue</testKey>");
    connection.send("test", "test", {testKey: "value"});
    connection.send("test", "test", ["value"]);
    connection.send("test", "test", "testValue");
    connection.close();
}

// Load the module
var rabbit = MooBot.loadModule('RabbitMQ');

// Create a new connection
var connection = rabbit.connect({
```

```
    brokers:[
       {
          host:"rabbithost",
          port:5672
       }
    ],
    user:"rabbitmq_admin",
    password:"78smr9!b",
    timeout:10000,
    vhost:"rabbitvhost",
    ssl:{
       ssl_protocol:"TLSv1.2",
       server_cert_file:"server.pem",
       client_cert_file:"client.pem",
       client_key_file:"client.key"
    }
});


if (connection) {
    // Send information
    connection.send("testExchange", "testRoutingKey", ["one", "two"]);

    // Close the connection
    connection.close();
}
```

## REST.V2

REST (Representational State Transfer) and RESTful applications use HTTP requests to post data (create and update), read data (make queries), and delete data.

The REST.V2 Moobot module accesses an external RESTful API through HTTP or HTTPS, offering consistent usage between the available methods and customization of HTTP requests sent.

It supports asynchronous operation (using callback functions) to send a request without blocking the JavaScript code execution until the request is completed. It supports use of the timeout property to make the request fail after a specified time.

REST.V2 is available to load into any standard Moobot.

To use, define a new global object REST at the top of a Moobot JavaScript file to load the module:

```
var REST = MooBot.loadModule('REST.V2');
```

### Reference Guide

#### REST.sendGet()

Sends a HTTP GET request to a third party (URL) with optional parameters:

Request Arguments

| Name | Type | Description |
|---|---|---|
| **url** | String | The request URL. Mandatory |
| **<parameters>** | JSON Object | Optional parameters. See below |

Optional parameters

| Name | Type | Description |
|---|---|---|
| **params** | String or Object | Either a String with the request encoded parameters or an Object with the parameters that will get encoded by the module. |

| | | |
|---|---|---|
| **user** | String | The user name for basic authentication. |
| **password** | String | The password for basic authentication. |
| **encrypted_password** | String | Encrypted version of password (encrypted using **moog_encryptor**). |
| **disable_certificate_validation** | Boolean | 'true' to disable HTTPS server certificate validation by the Moobot. |
| **headers** | Object | Any additional headers sent with the request. |
| **callback** | Callback function | The request is sent asynchronously, returns **null** and the **callback** function is called regardless of the success or failure of the request. See below. |
| **success** | Callback function | The request is sent asynchronously, returns **null** and the **success** function is called only the request was successful. See below. |
| **failure** | Callback function | The request is sent asynchronously, returns **null** and the **failure** function is called only if the request failed. See below. |
| **timeout** | Number | The period of time (in seconds) to wait for response before completing with timeout error.<br><br>If 0 or less, wait indefinitely. The default is 120 seconds. |
| **proxy** | String or Object | Host, port, user, encrypted_password/password.<br><br>For example, as an object:<br><br>**proxy:{**<br>**host:"proxyhost",**<br>**port:1223,**<br>**user:"proxyuser",**<br><br>**encrypted_password:"2KctaEbJH/m8rz4WqgmZYZfdripdIsku7fOFJWM6YNA="**<br>**//password: "unencrypted_plain_text_password"**<br>**}**<br><br>As an object, you can either specify a Cisco encrypted password or a plain text password, specifying both will favour the encrypted_password value.<br><br>Or, as a string, where format is <user>:<password>@<host>:<port><br><br>**proxy: "proxyuser:passw0rd@proxyhost:1223"**<br><br>Only plain text passwords are supported in the string format. |

Sending an asynchronous request (with callback functions)

To send a request without blocking the javascript code execution until the request is completed, define one (or more) of the callback functions: **callback**, **success** and **failure**. The REST.V2 module method (**send...**) then returns **null**, and sends the request in another thread.

Return Parameters

Sending a synchronous request returns a JavaScript object with the following fields:

| Name | Type | Description |
|---|---|---|
| **success** | Boolean | True if and only if the request was successful |
| **status_code** | Number | The HTTP status code of the request |

| | | |
|---|---|---|
| | | (200 = OK, 404 = Not found. Full list at w3.org) |
| **status_msg** | String | The message from the request ("OK", "Not found", etc.) |
| **response** | String | The response as raw text<br><br>Currently, binary response is not supported |
| **headers** | Object | The response HTTP headers |

Sending an asynchronous request (with callback functions) returns **null**. Once the request has completed, the callback function(s) are called with the reply object as the first (optional) parameter and the request object as the second (optional) parameter.

Examples

Each of the following gives details on the Cisco home page:

Synchronous request

```
var rc = REST.sendGet('http://www.Cisco.com');
```

Asynchronous request

```
function restSuccess(rc)
{
    var response = JSON.parse(rc.response);
    logger.info("number = " + response.records[0].number);
}

function restFailed(rc, req)
{
    var response = JSON.parse(rc.response);
    logger.info("URL:" + req.url +" failed - Msg:" + response.status_msg);
}

REST.sendGet({url: "http://www.Cisco.com",
                          success: restSuccess,
                          failure: restFailed});
```

Response

```
{
    "status_code": 200,
    "success": true,
    "response": "<!DOCTYPE html>... </body></html>",
    "status_msg": "OK",
    "headers": {
        "Transfer-Encoding": [
            "chunked"
        ],
        "Keep-Alive": [
            "timeout=15, max=100"
        ],
        "Server": [
            "Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.10 with Suhosin-
Patch mod_ssl/2.2.22 OpenSSL/1.0.1"
        ],
        "Connection": [
            "Keep-Alive"
        ],
        "Vary": [
            "Accept-Encoding"
        ],
        "Date": [
            "Fri, 30 Jan 2015 12:37:13 GMT"
```

```
            ],
            "Content-Type": [
                "text/html"
            ],
            "X-Powered-By": [
                "PHP/5.3.10-1ubuntu3.10"
            ]
        }
    }
```

### REST.sendPost()

Sends a HTTP POST request to a third party (URL) with optional parameters:

Request Arguments

| Name | Type | Description |
|---|---|---|
| **url** | String | The request URL. Mandatory |
| **<parameters>** | JSON Object | Optional parameters. See below |

Optional parameters

| Name | Type | Description |
|---|---|---|
| **params** | String or Object | Either a string with the request encoded parameters or an object with the parameters that will get encoded by the module. |
| **content_type** | String | The content type of the body. |
| **body** | String or Object | The request body. Either a string (that will be sent as is) or an object. If the **content_type** is “application/json” and the body is an object, the body will be sent as JSON. Otherwise it will be sent as URL encoded. |
| **user** | String | The user name for basic authentication. |
| **password** | String | The password for basic authentication. |
| **encrypted_password** | String | Encrypted version of password (encrypted using **moog_encryptor**). |
| **disable_certificate_validation** | Boolean | 'true' to disable HTTPS server certificate validation by the Moobot. |
| **headers** | Object | Any additional headers sent with the request. |
| **callback** | Callback function | The request is sent asynchronously, returns **null** and the **callback** function is called regardless of the success or failure of the request. See below. |
| **success** | Callback function | The request is sent asynchronously, returns **null** and the **success** function is called only the request was successful. See below. |
| **failure** | Callback function | The request is sent asynchronously, returns **null** and the **failure** function is called only if the request failed. See below. |
| **timeout** | Number | The period of time (in seconds) to wait for response before completing with timeout error.<br><br>If 0 or less, wait indefinitely. The default is 120 seconds. |
| **proxy** | String or Object | Host, port, user, encrypted_password/password.<br><br>For example, as an object:<br><br>**proxy:{** |

| | | |
|---|---|---|
| | | **host:"proxyhost",**<br>**port:1223,**<br>**user:"proxyuser",**<br><br>**encrypted_password:"2KctaEbJH/m8rz4WqgmZYZfdripdIsku7fOFJWM6YNA="**<br>**//password: "unencrypted_plain_text_password"**<br>**}**<br><br>As an object, you can either specify a Cisco encrypted password or a plain text password, specifying both will favour the encrypted_password value.<br><br>Or, as a string, where format is <user>:<password>@<host>:<port><br><br>**proxy: "proxyuser:passw0rd@proxyhost:1223"**<br><br>Only plain text passwords are supported in the string format. |

Sending an asynchronous request (with callback functions)

To send a request without blocking the JavaScript code execution until the request is completed, define one (or more) of the callback functions: **callback**, **success** and **failure**. The REST.V2 module method (**send...**) then returns **null**, and sends the request in another thread.

Return Parameters

Sending an asynchronous request (with Callback functions) returns **null**. See above.

Sending a synchronous request returns a JavaScript object with the following fields:

| Name | Type | Description |
|---|---|---|
| **success** | Boolean | True if and only if the request was successful. |
| **status_code** | Number | The HTTP status code of the request.<br><br>(200 = OK, 404 = Not found. Full list at w3.org) |
| **status_msg** | String | The message from the request ("OK", "Not found", etc.) |
| **response** | String | The response as raw text. Binary response is not supported. |
| **headers** | Object | The response HTTP headers. |

Sending an asynchronous request (with callback functions) returns **null**. Once the request has completed, the callback function(s) are called with the reply object as the first (optional) parameter and the request object as the second (optional) parameter.

Examples

Each of the following accesses DuckDuckGo and searches for 'Cisco'.

Synchronous request:

```
var rc = REST.sendPost('https://api.duckduckgo.com/',
{q:'Cisco', format:'json', pretty:1});
```

Asynchronous request:

```
REST.sendPost({url: 'https://api.duckduckgo.com/',
   body: {q:'Cisco', format:'json', pretty:1},
   timeout: 4.2,
   callback: function(rc) {
   ...
   }});
```

Here, the request has a timeout set of 4.2 seconds.

Responses

For the synchronous request, and for the asynchronous request if it doesn't time out:


```
{
    "status_code": 200,
    "success": true,
    "response": "{   \"DefinitionSource\" : \"\",   \"Heading\" : \"\",
\"ImageWidth\" : 0,   ... : \"\"}",
    "status_msg": "OK",
    "headers": {
        "Transfer-Encoding": [
            "chunked"
        ],
        "Strict-Transport-Security": [
            "max-age=0"
        ],
        "Cache-Control": [
            "max-age=1"
        ],
        "Server": [
            "nginx"
        ],
        "X-DuckDuckGo-Results": [
            "1"
        ],
        "X-DuckDuckGo-Locale": [
            "en_US"
        ],
        "Connection": [
            "keep-alive"
        ],
        "Expires": [
            "Fri, 30 Jan 2015 12:44:47 GMT"
        ],
        "Date": [
            "Fri, 30 Jan 2015 12:44:46 GMT"
        ],
        "Content-Type": [
            "application/x-javascript"
        ]
    }
}
```


...if the asynchronous request times out:


```
{
    "status_code": 408,
    "success": false,
    "status_msg": "Request Time-Out"
}
```


### REST.sendPut()

Sends a HTTP PUT request to a third party (URL) with optional parameters:

Request Arguments

| Name | Type | Description |
|---|---|---|
| **url** | String | The request URL. Mandatory. |
| **<parameters>** | JSON Object | Optional parameters. See below. |

Optional parameters

| Name | Type | Description |
|---|---|---|
| **params** | String or Object | Either a string with the request encoded parameters or an object with the parameters that will get encoded by the module. |
| **content_type** | String | The content type of the body. |
| **body** | String or Object | The request body. Either a string (that will be sent as is) or an object. If the **content_type** is “application/json” and the body is an object, the body will be sent as JSON. Otherwise it will be sent as URL encoded. |
| **user** | String | The user name for basic authentication. |
| **password** | String | The password for basic authentication. |
| **encrypted_password** | String | Encrypted version of password (encrypted using **moog_encryptor**). |
| **disable_certificate_validation** | Boolean | 'true' to disable HTTPS server certificate validation by the Moobot. |
| **headers** | Object | Any additional headers sent with the request. |
| **callback** | Callback function | The request is sent asynchronously, returns **null** and the **callback** function is called regardless of the success or failure of the request. See below. |
| **success** | Callback function | The request is sent asynchronously, returns **null** and the **success** function is called only the request was successful. See below. |
| **failure** | Callback function | The request is sent asynchronously, returns **null** and the **failure** function is called only if the request failed. See below. |
| **timeout** | Number | The period of time (in seconds) to wait for response before completing with timeout error.<br><br>If 0 or less, wait indefinitely. The default is 120 seconds. |

Sending an asynchronous request (with callback functions)

To send a request without blocking the JavaScript code execution until the request is completed, define one (or more) of the callback functions: **callback**, **success** and **failure**. The REST.V2 module method (**send...**) then returns **null**, and sends the request in another thread.

Return Parameters

Sending a synchronous request returns a JavaScript object with the following fields:

| Name | Type | Description |
|---|---|---|
| **success** | Boolean | True if and only if the request was successful. |
| **status_code** | Number | The HTTP status code of the request.<br><br>(200 = OK, 404 = Not found. Full list at w3.org) |
| **status_msg** | String | The message from the request ("OK", "Not found", etc.) |
| **response** | String | The response as raw text.<br><br>Binary response is not supported. |
| **headers** | Object | The response HTTP headers. |

Sending an asynchronous request (with callback functions) returns **null**. Once the request has completed, the callback function(s) are called with the reply object as the first (optional) parameter and the request object as the second (optional) parameter.

Example

The following stores the specified information at the URL (similar to a file upload):

Request

```
var rc = REST.sendPut('http://api.acme.com/reportIncident',
'{"incident":"broken fan","location":"office2"}');
```

Response

```
{
    "status_code": 204,
    "success": true,
    "response": "",
    "status_msg": "No Content",
    "headers": {
        "Connection": [
            "keep-alive"
        ],
        "Date": [
            "Fri, 30 Jan 2015 12:55:59 GMT"
        ]
    }
}
```

When POSTing or PUTting URL encoded data (a content-type of "application/x-www-form-urlencoded") complex objects will need to be either split into individual key:value pairs suitable for url encoding or simply JSON stringify the object in its entirety. Stringifying the object will require the receiver to be able to parse the string value back to an object if needed. If the receiver cannot do this parsing then the object will need to be broken into key value pairs. For example, to send the entire alert custom_info object as part of a URL-encoded body:

```
var custom_info = alert.getCustomInfo();
var payload;
try {
payload = JSON.stringify(custom_info);
}
catch(e) {
logger.info("Failed to stringify custom_info " + e );
payload = null;
}


        var postParams={
            "url" : "http://www.someurl.com/someEndpoint",
            "body" : payload,
            "content_type" : "application/x-www-form-urlencoded"
};

var request = rest.sendPost(postParams);
```

### REST.sendDelete()

Sends an HTTP DELETE request to a third party (URL) with optional parameters:

Request Arguments

| Name | Type | Description |
|---|---|---|
| **url** | String | The request URL. Mandatory. |
| **<parameters>** | JSON Object | Optional parameters. See below. |

Optional parameters

| Name | Type | Description |
|---|---|---|

| | | |
|---|---|---|
| **params** | String or Object | Either a string with the request encoded parameters or an object with the parameters that will get encoded by the module. |
| **user** | String | The user name for basic authentication. |
| **password** | String | The password for basic authentication. |
| **encrypted_password** | String | Encrypted version of password (encrypted using **moog_encryptor**). |
| **disable_certificate_validation** | Boolean | 'true' to disable HTTPS server certificate validation by the Moobot. |
| **headers** | Object | Any additional headers sent with the request. |
| **callback** | Callback function | The request is sent asynchronously, returns **null** and the **callback** function is called regardless of the success or failure of the request. See below. |
| **success** | Callback function | The request is sent asynchronously, returns **null** and the **success** function is called only the request was successful. See below. |
| **failure** | Callback function | The request is sent asynchronously, returns **null** and the **failure** function is called only if the request failed. See below. |
| **timeout** | Number | The period of time (in seconds) to wait for response before completing with timeout error.<br><br>If 0 or less, wait indefinitely. The default is 120 seconds. |

Sending an asynchronous request (with callback functions)

To send a request without blocking the javascript code execution until the request is completed, define one (or more) of the callback functions: **callback**, **success** and **failure**. The REST.V2 module method (**send...**) then returns **null**, and sends the request in another thread.

Return Parameters

Sending a synchronous request returns a JavaScript object with the following fields:

| Name | Type | Description |
|---|---|---|
| **success** | Boolean | True if and only if the request was successful. |
| **status_code** | Number | The HTTP status code of the request.<br><br>(200 = OK, 404 = Not found. Full list at w3.org) |
| **status_msg** | String | The message from the request ("OK", "Not found", etc.) |
| **response** | String | The response as raw text.<br><br>Binary response is not supported. |
| **headers** | Object | The response HTTP headers. |

Sending an asynchronous request (with callback functions) returns **null**. Once the request has completed, the callback function(s) are called with the reply object as the first (optional) parameter and the request object as the second (optional) parameter.

Example

The following sends a delete request to the specified URL, with additional headers criteria:

Request:

```
var rc =
REST.sendDelete({url:"http://moogbox2:9090/deletePassport/123456789","head
```

```
ers":{"user-agent":"moobot","accept":"text/plain","accept-language":"en-
US"}});;
```

Response

```
{
    "status_code": 200,
    "success": true,
    "response": "{\t\"remoteId\": 33,\t\"weight\":
0.8240487528964877,\t\"location\": {\t\t\"latitude\":
147.3387699946761,\t\t\"longitude\": -7.957067163661122\t}}",
    "status_msg": "OK",
    "headers": {
        "Transfer-Encoding": [
            "chunked"
        ],
        "Connection": [
            "keep-alive"
        ],
        "Date": [
            "Fri, 30 Jan 2015 12:49:44 GMT"
        ],
        "Content-Type": [
            "application/json"
        ]
    }
}
```

### REST.send()

A generic send request for sending other HTTP methods as part of the request properties (**'GET'**, **'HEAD',** etc.). Optional parameters for synchronous and asynchronous requests are available as described in the above methods.

Example

The following returns time/date information from the Cisco server:

Request

```
var rc = REST.send({method: 'HEAD', url: 'http://www.Cisco.com/'});
logger.warning("rc: " + JSON.stringify(rc, null, "\t") );
var date = rc.headers.Date[0];
logger.warning("date " + date );
```

Response

```
{
    "status_code": 204,
    "success": true,
    "response": "",
    "status_msg": "OK",
    "headers": {
        "Keep-Alive": [
            "timeout=15, max=100"
        ],
        "Server": [
            "Apache/2.2.22 (Ubuntu) PHP/5.3.10-1ubuntu3.10 with Suhosin-
Patch mod_ssl/2.2.22 OpenSSL/1.0.1"
        ],
        "Connection": [
            "Keep-Alive"
        ],
        "Vary": [
            "Accept-Encoding"
        ],
```

```
            "Date": [
                "Fri, 30 Jan 2015 13:00:33 GMT"
            ],
            "Content-Type": [
                "text/html"
            ],
            "X-Powered-By": [
                "PHP/5.3.10-1ubuntu3.10"
            ]
        }
    }
```

## Proxy Examples

The following examples show how to configure Cisco Crosswork Situation Manager Moobots when a proxy server is used for connection to Cisco.

You can define a proxy in the following ways:

```
proxy: "proxyuser:passw0rd@proxyhost:1223"

proxy: "proxyhost:1223"

proxy: {
        host: "proxyhost",
        port: 1223
        }
```

Situation Manager

The following example shows how to update the Situation Manager to send a REST.V2 updateSituation message through a proxy server.

1. Edit the Situation Manager Moobot file, located at **$MOOGSOFT_HOME/bots/moobots/SituationMgr.js**.

2. Modify the **updateSitn** function to utilize the POST action. For example:

```
function updateSitn(situation)
{
        var sig_id = situation.value("sig_id");
        logger.warning("Update Situation Processed: " + sig_id);
    doPOST(sig_id);
}
```

3. Insert the proxy block into the **POST** action. For example:

```
function doPOST(sig_id)
{
        var request = REST.sendPost({
                url:"http://surveilanceserver_84:9090/reportAntiSoc",
                params: {
                        crime: "Graffiti"
                },
        proxy: {
            host: "proxyserver",
            port : 3128,
            user : "username",

encrypted_password:"zm0lxjTGiAhp6LrpM49+kr4SDtHj/fq16+i+hD1MG4c="
        },
                callback: function(response, request)
                {
```

```
                        if (response.success) {
                                logger.warning("4764 CALLBACK SUCCESS
("+sig_id+") RESPONSE - ("+response.status_code +" -
"+response.response+") REQUEST - "+ JSON.stringify(request));
                        } else {
                                logger.warning("4764 CALLBACK FAILURE
("+sig_id+") RESPONSE - ("+response.status_code +" - "+response.response+"
- "+response.status_msg+") REQUEST -
                                " + request.status_code + " " +
request.response + " " + request.status_msg);
                        }
                }
        });
        logger.warning("4764 POST REQUEST SENT FOR "+sig_id+" ...");
}
```

ServiceNow

The following example demonstrates how to configure the ServiceNow ticketing integration when Cisco Crosswork Situation Manager is installed on-prem and ServiceNow is in the cloud, and the two systems communicate through a proxy server.

1. Edit the ServiceNow Moobot file, located at **`$MOOGSOFT_HOME/bots/moobots/ServiceNow-2.0-Geneva.js`** and define a variable containing the proxy details. For example:

```
var proxy = {
  host: 'proxy-app.company.com',
  timeout:60,
  port: 8080
  }
```

2. Add the proxy to the POST actions in the **`AddToWorkNotes`** and **`resolveIncident`** functions. For example:

```
var rc = REST.sendPost({
  'url': url,
  'body': JSON.stringify(urlParameters),
  'user': user,
  'password': password,
  'content_type': "application/json",
  'proxy': proxy,
  'disable_certificate_validation': true
  });
```

## Utilities

The Utilities module is a JavaScript utility that allows you to escape XML so that Cisco Crosswork Situation Manager correctly interprets control characters as data, not markup.

You can also use the module to convert an XML string to a JSON object, which is easier to manipulate in JavaScript. You can convert a JSON object to XML for external communication that requires XML input.

### Load the Utilities Module

You can load the Utilities module into any standard Moobot or LAMbot.

To use, define a global object **`utilities`** at the top of a Moobot or LAMbot js file to load the Utilities module:

Moobots

```
var utilities = MooBot.loadModule('Utilities');
```

```
var utilities = LamBot.loadModule('Utilities');
```

## Command Reference

### utilities.escapeXML()

Escapes an XML string. Certain characters will not parse correctly if they are not escaped:

| Unescaped character | Escaped string |
|---|---|
| " | " |
| ' | ' |
| < | < |
| > | > |
| & | & |

Request Argument

| Name | Type | Description |
|---|---|---|
| **value** | String | The string to escape. |

Example

```
var unescapedXML = 'my content requires "< and > "';
var escapedXML = '<tag>' + utilities.escapeXML(unescapedXML) + '</tag>';
```

The variable **escapedXML** now contains:

```
<tag>my content requires &quot;&lt; and &gt; &quot;</tag>
```

### utilities.unescapeXML()

Unescapes an XML string.

| Name | Type | Description |
|---|---|---|
| **value** | String | The string to unescape. |

Example

```
var escapedXML = '<tag>my content requires &quot;&lt; and &gt;
&quot;</tag>';
var unescapedXML = utilities.unescapeXML(escapedXML);
```

The variable **unescapedXML** now contains:

```
<tag>my content requires "< and > "</tag>
```

### utilities.xmlToJSON()

Converts an XML string to a JSON object.

| Name | Type | Description |
|---|---|---|
| **value** | XML string | The XML to convert to JSON. |

Example

```
var xmlExample = '<alerts>' +
        '<alert enriched="false">' +
        '<id>1</id>' +
        '<description>Alert 1</description>' +
        '<host>email.moogsoft.com</host>' +
        '<severity>5</severity>' +
        '</alert>' +
```

```
        '<alert enriched="true">' +
        '<id>2</id>' +
        '<description>Alert 2</description>' +
        '<host>calendar.moogsoft.com</host>' +
        '<severity>2</severity>' +
        '</alert>' +
        '</alerts>';

var alerts = utilities.xmlToJSON(xmlExample);
```

The variable **alerts** now contains:

```
{
  "alerts":{
    "alert":
    [
      {
        "severity":5,
        "host":"email.moogsoft.com",
        "description":"Alert 1",
        "id":1,
        "enriched":false
      },
      {
        "severity":2,
        "host":"calendar.moogsoft.com",
        "description":"Alert 2",
        "id":2,
        "enriched":true
      }
    ]
  }
}
```

#### utilities.jsonToXML

Converts a JSON object to an XML string. You can only use the utility to convert JSON objects, not arrays.

| Name | Type | Description |
|---|---|---|
| **value** | JSON object | The JSON object to convert to XML. |

Example

```
var jsonObjectExample =
{
  "data": {
    "alerts":
    [
      {
         "enriched": "false",
         "id": "1",
         "description": "Alert 1",
         "host": "email.moogsoft.com",
         "severity": "5"
      },
      {
         "enriched": "true",
         "id": "2",
         "description": "Alert 2",
         "host": "calendar.moogsoft.com",
         "severity": "2"
      }
    ]
```

```
    }
};
```

```
var convertedXML = utilities.jsonToXML(jsonObjectExample);
```

The variable **convertedXML** now contains:

```
<data>
   <alerts>
      <severity>5</severity>
      <enriched>false</enriched>
      <host>email.moogsoft.com</host>
      <description>Alert 1</description>
      <id>1</id>
   </alerts>
   <alerts>
      <severity>2</severity>
      <enriched>true</enriched>
      <host>calendar.moogsoft.com</host>
      <description>Alert 2</description>
      <id>2</id>
   </alerts>
</data>
```

# Programmatic LAM

The Programmatic LAM is a custom polling LAM. It is an advanced version of the REST Client LAM. The REST Client LAM accepts a single API call and parses the responses it receives into Cisco Crosswork Situation Manager events. The Programmatic LAM can accept multiple calls but you must define the processing yourself in the LAMbot using JavaScript.

## Before You Begin

Before you start to configure the LAM, ensure you have met the following requirements:

You have the details of the API to query.

You can write JavaScript.

## Configure the LAM

Edit the configuration file to control the behavior of the Programmatic LAM. You can find the file at **$MOOGSOFT_HOME/config/programmatic_lam.conf**.

- Configure the behavior of the LAM:
  - request_interval: Length of time to wait between calls to the execute method, in seconds. Defaults to 60.
  - num_threads: Number of worker threads to use for processing events. Defaults to 5.
- Optionally configure the LAM identification and logging details in the agent and log_config sections of the file:
  - name: Identifies events the LAM sends to the Message Bus.
  - capture_log: Name and location of the LAM's log file.
  - configuration_file: Name and location of the LAM's process log configuration.

## Example LAM Configuration

An example Programmatic LAM configuration is as follows:

```
monitor:
{
```

```
    name                : "Programmatic LAM",
    request_interval       : 60,
    num_threads         : 5
    agent:
    {
        name                : "ProgrammaticLam",
        capture_log         : "$MOOGSOFT_HOME/log/data-
capture/programmatic_lam.log"
    },
    log_config:
    {
        configuration_file :
"$MOOGSOFT_HOME/config/logging/custom.log.json"
    }
}
```

## Configure the LAMbot

You must configure the Programmatic LAMbot with JavaScript code to process and filter events and send them to the Message Bus.

You can find the LAMbot file at **`$MOOGSOFT_HOME/bots/lambots/ProgrammaticLam.js`**.It contains the following functions.

onLoad

The LAMbot calls the onLoad function when it is first initialized. Use it to set up any structures and variables required for subsequent processing.

execute

The execute function takes an argument, programmaticApi. It allows you to pass state information from one execute call to another. For example, if you are polling an endpoint that requires a time variable, you can pass the last time value so that the next poll can start from that value.

The state is saved to the MoogDb database for use during failover from active to passive in a HA environment. When passive becomes active the LAMbot reads the state from the database and uses the correct information in its next poll.

The execute function calls the following modules:

REST.V2: Use this module to query an external endpoint. See REST.V2 for more information.

ExternalDb: Use this module to execute queries on databases that support JDBC connections. See ExternalDb for more information.

The execute function contains the following methods:

getState: Allows you to pass state information from one execute call to another. State is automatically set by the return object of the execute function call. For example: **`return { events [ ], state { } };`**

captureLog: Allows you to write raw event data to the log file defined in the capture_log property in the LAM's configuration file.

Example Return Object

An example return object from the execute function containing an event with description, class and host information is as follows:

```
return {
    "events": [ { "description":"Loss of Signal","class":"Gigabit
Ethernet","host":"S-CARP282" } ],
    "state": { "last_poll_time": 649077928 }
};
```

presend

The LAMbot calls the presend function every time it assembles an event to publish on the Message Bus. If the function returns true, the event is published on the bus. If it returns false, the event is discarded. Moogfarmd processes published events and turns them into alerts and Situations.

Use the presend function to define the conditions in which events will and will not be published. You can also use the function to partition event streams for differential processing in a distributed environment.