



Cisco Crosswork Situation Manager 7.3.x Developer Guide

(Powered by Moogsoft AIOps 7.3)

Table of Contents

Developer Guide	5
Graze API	5
Stats API.....	5
Moobot Modules	6
Programmatic LAM.....	6
Graze API	6
Endpoints	6
Before you begin	6
API definition.....	7
Authentication.....	7
POST parameters.....	7
Graze API EndPoint Reference	8
Integrations API.....	190
Alert Action Codes.....	206
Situation Action Codes	207
Situation Flags	208
API Update Behavior	209
Stats API	209
Integrations API	293
Endpoints	293
API definition.....	293
Authentication.....	293
Integrations API Endpoint Reference	293
/integrations.....	294
/integrations/{integrationId}.....	298
/integrations/{integrationId}/status	306
Export and Import Integrations.....	308

MoogDb V2	313
Load MoogDb V2	313
Methods	313
Methods	313
checkSituationFlag	355
deleteTeam	356
getProcesses.....	357
getServices	358
getSituationFlags	359
getSituationPrimaryTeam	360
getSituationsWithFlag	361
getToolShares.....	362
getTopPrcDetails	362
rateSituation.....	364
removeSituationPrimaryTeam	365
setSituationFlags	366
setSituationPrimaryTeam.....	367
shareToolAccess	368
Moobot Modules	369
Threads and Global Scope	369
Moobot Modules.....	369
Examples	370
Using External Modules.....	371
onLoad Function.....	371
Config	372
Constants.....	373
Events	376
Expose Active Moolets	389

ExternalDb	390
Graph Topology	398
Logger	404
Mailer	406
Moolet Informs	408
Moolet Information API	411
Process	413
RabbitMQ	415
REST.V2.....	419
Utilities	431
Programmatic LAM.....	434
Before You Begin	434
Configure the LAM	435
Example LAM Configuration.....	435
Configure the LAMbot	435

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 create, retrieve, and update data 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 Moollets.
- 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 [Enrichment](#) 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. See [Integrations](#) for available ticketing integrations in Cisco Crosswork Situation Manager.

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](#) for more installation and configuration instructions. Configure Grafana Example

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

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

Use caution when employing the Graze API. Excessive requests can impact overall system performance, especially **getSituationIds** and **getAlertIds**.

Contact Cisco Support if you experience difficulties or need further guidance.

Endpoints

See [Graze API EndPoint Reference](#) for details of all the Graze API endpoints.

Before you begin

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

Configure Apache Tomcat

You must configure Apache Tomcat to allow cross-context calls to be made by adding the following to the **context.xml** file in the Apache 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
```

Authentication

All requests, other than **authenticate**, require a basic authentication header or a valid **auth_token**. You must make a valid **authenticate** request before using any Graze API request without a basic authentication header.

If you make regular Graze requests within a one hour timeframe, you are considered active and your session does not expire. Inactive sessions are logged out after one hour, and you must make a new **authenticate** request to get a new valid **auth_token**.

Authentication troubleshooting

If an error occurs during Graze login authentication, Cisco Crosswork Situation Manager returns the following output:

```
{"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 about 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:

- For example, the user is not assigned the Grazer role:

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

- For example, no user of that name:

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

- For example, incorrect password:

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

POST parameters

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

form-urlencoded

To send POST parameters as **form-urlencoded** parameters, set the content type to **application/x-www-form-urlencoded**. If the character set is not set, UTF-8 is assumed.

Example cURL command:

```
"https://localhost/graze/v1/resolveSituation?auth_token=b40244fd79aa46fba76c60c56d538c49&sitn_id=10" --insecure -X POST -v
```

application/json

To supply POST parameters as JSON within the body of the request, set the content type to **application/json**. If the character set is not set, UTF-8 is assumed.

Example 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}'
```

Graze API EndPoint Reference

This is a reference list for the Graze API endpoints. Follow the links to see the details of each endpoint.

Alerts

The following endpoints relate to alerts:

- [addAlertCustomInfo](#): Adds and merges custom information for a specified alert.
- [addAlertToSituation](#): Adds a specified alert to a specified Situation.
- [assignAlert](#): Assigns the specified user as the owner of a specified alert.
- [assignAndAcknowledgeAlert](#): Assigns and acknowledges the specified user as the owner of a specified alert.
- [closeAlert](#): Closes one or more alerts.
- [deassignAlert](#): Deassigns the current owner from a specified alert.
- [getAlertActions](#): Returns the actions for one or more alerts.
- [getAlertDetails](#): Returns details, such as the description or severity, of a specified alert.
- [getAlertIds](#): Returns the total number of alerts, and a list of the alert IDs, for a specified alert filter and a limit.
- [removeAlertFromSituation](#): Removes a specified alert from a specified Situation.
- [resolveAlerts](#): Resolves a list of alerts.
- [setAlertAcknowledgeState](#): Acknowledges or unacknowledges the owner of a specified alert.
- [setAlertSeverity](#): Sets the severity level of a specified alert.

Algorithms

The following endpoints relate to Cookbooks and Recipes:

- [addBotRecipe](#): Creates a new Cookbook Bot Recipe.

- [addCookbook](#): Creates a new Cookbook.
- [addValueRecipe](#): Creates a new Cookbook Recipe using Value Recipe or Value Recipe v2 recipe types.
- [deleteCookbook](#): Deletes an existing Cookbook.
- [deleteRecipe](#): Deletes an existing Cookbook Recipe.
- [getCookbooks](#): Returns all the Cookbooks in Cisco Crosswork Situation Manager.
- [getRecipes](#): Returns all the Recipes in Cisco Crosswork Situation Manager.
- [updateBotRecipe](#): Updates a Cookbook Bot Recipe.
- [updateCookbook](#): Updates a Cookbook.
- [updateValueRecipe](#): Updates a Cookbook Recipe that uses either a Value Recipe or a Value Recipe v2 recipe type.

The following endpoints relate to merge groups:

- [addMergeGroup](#): Adds a new custom merge group.
- [deleteMergeGroup](#): Deletes an existing custom merge group.
- [getDefaultMergeGroup](#): Returns details of the default merge group in Cisco Crosswork Situation Manager.
- [getMergeGroups](#): Returns details of all the custom merge groups in Cisco Crosswork Situation Manager.
- [updateDefaultMergeGroup](#): Updates the default merge group in Cisco Crosswork Situation Manager.
- [updateMergeGroup](#): Updates a custom merge group.
- The following endpoints relate to Tempus:
- [addTempus](#): Adds a new Tempus Moolet.
- [deleteTempus](#): Deletes an existing Tempus Moolet.
- [getTempus](#): Returns the details of all Tempus Moolets in Cisco Crosswork Situation Manager.
- [updateTempus](#): Updates an existing Tempus Moolet.

Dashboards and Reporting

See the [Stats API](#) for information on endpoints that provide statistics related to dashboards or reporting.

Processes and Configuration

The following endpoints relate to Cisco Crosswork Situation Manager processes and configuration:

- [addProcess](#): Adds a new process to the database.
- [addService](#): Adds a new external service to the database.
- [createMaintenanceWindow](#): Creates a maintenance window that filters alerts caused by a known period of maintenance.
- [deleteMaintenanceWindow](#): Deletes a single maintenance window.

- [deleteMaintenanceWindows](#): Deletes maintenance windows that match a specified filter.
- [findMaintenanceWindows](#): Returns maintenance windows that match a specified filter.
- [getIntegrationConfig](#): Exports the configuration and mapping needed for an integration in JSON format.
- [getMaintenanceWindows](#): Returns maintenance windows based on the window ID and how many should be fetched.
- [getProcesses](#): Returns a list of the processes in the database.
- [getServices](#): Returns a list of the services in the database.
- [getSeverities](#): Returns a list of possible severities and their severity IDs.
- [getStatuses](#): Returns a list of statuses that can apply to Situations and their IDs.
- [getSystemStatus](#): Returns current system status information for all processes.
- [getSystemSummary](#): Returns a summary of current alerts and Situations in Cisco Crosswork Situation Manager.
- [getToolShares](#): Returns the shared access for a specified tool.
- [shareToolAccess](#): Shares access to a tool with other users, teams, or roles, or makes it global so that all users can access it.
- [updateMaintenanceWindow](#): Updates an existing maintenance window.

Situations

The following endpoints relate to Situations:

- [addSigCorrelationInfo](#): Associates the external client with a specified Situation.
- [addSituationCustomInfo](#): Adds and merges custom information for a specified Situation.
- [addThreadEntry](#): Adds a new entry to an existing thread in a Situation.
- [assignAndAcknowledgeSituation](#): Assigns and acknowledges the moderator to the Situation for a specified situation ID and user ID.
- [assignSituation](#): Assigns the moderator to the Situation for a specified Situation ID and user ID.
- [assignTeamsToSituation](#): Assigns one or more teams to a Situation, or unassigns all teams from a Situation.
- [closeSituation](#): Closes a specified Situation which is currently open, and optionally closes alerts in the Situation.
- [createSituation](#): Creates a manual Situation. The Situation description is set with the **description** parameter.
- [createThread](#): Creates a new thread for a specified Situation.
- [createThreadEntry](#): Creates a new entry in an existing thread in a Situation. This endpoint has been superseded by [addThreadEntry](#).
- [deassignSituation](#): Deassigns the current moderator from the Situation for a specified Situation ID.

- [getActiveSituationIds](#): Returns the total number of active Situations, and a list of their Situation IDs.
- [getPrcLabels](#): Returns probable root cause (PRC) information for all alerts or specified alerts within a Situation.
- [getResolvingThreadEntries](#): Returns thread entries for a specified Situation that have been marked as resolving steps.
- [getSigCorrelationInfo](#): Returns all correlation information related to a specified Situation.
- [getSimilarSituationIds](#): Returns a list of IDs of similar Situations, for a specified Situation ID and a limit.
- [getSimilarSituations](#): Returns the details of similar Situations for a specified Situation and a limit.
- [getSituationActions](#): Returns the actions for a list of Situations.
- [getSituationAlertIds](#): Returns the total number of alerts, and a list of the alert IDs for a specified Situation.
- [getSituationDescription](#): Returns the description for a specified Situation.
- [getSituationDetails](#): Returns the details of a specified Situation.
- [getSituationHosts](#): Returns the hosts for a specified Situation.
- [getSituationIds](#): Returns the total number of Situations, and a list of their Situation IDs, for a specified Situation filter and a limit.
- [getSituationPrimaryTeam](#): Returns the primary team on the specified Situation.
- [getSituationProcesses](#): Returns a list of process names for a specified Situation.
- [getSituationServices](#): Returns a list of external service names for a specified Situation.
- [getSituationTopology](#): Returns a JSON object that represents the nodes affected by the Situation.
- [getSituationVisualization](#): Returns information on the origin and cause of a specified Situation.
- [getThreadEntries](#): Returns thread entries for a specified Situation.
- [getTopPrcDetails](#): Returns the top most likely causal alerts, based on their Probable Root Cause value, for a specified Situation.
- [mergeSituations](#): Merges multiple specified Situations.
- [rateSituation](#): Applies a rating to a specified Situation.
- [removeSigCorrelationInfo](#): Removes all correlation information related to a specified Situation.
- [removeSituationPrimaryTeam](#): Removes the primary team from a Situation.
- [resolveSituation](#): Resolves a specified Situation that is currently open.
- [setPrcLabels](#): Sets the probable root cause (PRC) labels for specified alerts within a Situation.
- [setResolvingThreadEntry](#): Sets or clears a thread entry in a Situation as a resolving step.

- [setSituationAcknowledgeState](#): Acknowledges or unacknowledges the moderator to the Situation for a specified Situation ID and acknowledged state.
- [setSituationDescription](#): Sets the description for a specified Situation.
- [setSituationPrimaryTeam](#): Sets one of the teams already assigned to a Situation as the primary team.
- [setSituationProcesses](#): Applies a list of processes to a specified Situation.
- [setSituationServices](#): Applies a list of external services to a specified Situation.

Security Realms

The following endpoints relate to security realms:

- [createSecurityRealm](#): Creates a new security realm from an Identity Provider (IdP) URL.
- [getSecurityRealm](#): Returns a JSON object containing the names and configuration details of active security realms.
- [updateSecurityRealm](#): Updates an existing security realm in the database.

User Management

The following endpoints relate to the management of users, teams and roles:

- [applyNewLicense](#): Adds a Cisco Crosswork Situation Manager license via Graze.
- [authenticate](#): Provides the auth_token required by all other Graze API requests which do not provide the basic authentication header.
- [createTeam](#): Creates a new team.
- [createUser](#): Creates a new user.
- [deleteTeam](#): Deletes a single team.
- [getTeam](#): Returns a team's details by team ID or name.
- [getTeamsForService](#): Returns all teams related to the service with the specified ID or name.
- [getTeamSituationIds](#): Returns the total number of Situations that are assigned to a team, and a list of their Situation IDs.
- [getUserInfo](#): Returns information about a specified user.
- [getUserRoles](#): Returns the user's roles from the database.
- [getUsers](#): Returns a list of all users in the database.
- [getUserTeams](#): Returns the team names and IDs associated with the specified user ID or username.
- [updateTeam](#): Updates an existing team.
- [updateUser](#): Updates an existing user.

Workflow Engine

The following endpoints relate to the Workflow Engine:

- [createWorkflow](#): Creates a new workflow in the Workflow Engine.

- [deleteWorkflow](#): Deletes a workflow from the Workflow Engine.
- [getWorkflowEngineMoolets](#): Returns a list of all the workflows in all the Workflow Engine Moolets in Cisco Crosswork Situation Manager.
- [getWorkflows](#): Returns workflows for a specified Workflow Engine Moolet.
- [reorderWorkflows](#): Reorders the sequence of workflows within a Workflow Engine Moolet.
- [updateWorkflow](#): Updates an existing workflow in the Workflow Engine.

addAlertCustomInfo

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addAlertCustomInfo** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	Alert ID.
custom_info	JSON object	Yes	A JSON object containing the custom information.

Response

Endpoint **addAlertCustomInfo** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	No
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **addAlertCustomInfo**:

Request example

Example cURL request to add custom info to "field1", "field2", "field3", and "field4" in alert ID 9:

```
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"} }{'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addAlertToSituation

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addAlertToSituation** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	Alert ID.
sitn_id	Numnber	Yes	Situation ID.

Response

Endpoint **addAlertToSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

This endpoint does not add the alert to the Situation if the alert has been archived to the historic database even if the Situation is still in the active database.

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **addAlertToSituation**:

Request example

Example cURL request to add alert ID 16 to Situation ID 7:

```
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 }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addBotRecipe

A Graze API POST request that creates a new Cookbook Bot Recipe. To create Recipes using the Value Recipe and Value Recipe v2 recipe types, use [addValueRecipe](#).

See </document/preview/114188#UUID1ecd7d867ea666f30327761f670221b7> for more information. Recipe Types

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addBotRecipe** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
cookbooks	Array of Strings	No	A list of the Cookbooks that this Recipe belongs to. You can add Cookbooks here or, when you create a Cookbook, you can assign the Recipes to it.
name	String	Yes	Name of the Recipe. Use a unique and descriptive name.
description	String	No	Description of the Recipe. Default is the Recipe name .
alert_threshold	Positive Integer	No	Minimum number of alerts required before Cookbook creates a Situation.
trigger	String	No	A filter that determines the alerts that Cookbook considers for Situation creation. Cookbook includes alerts that match the trigger filter. By default Cookbook only includes alerts with a severity of 'Critical'. For details on creating a filter, see /document/preview/35090#UUIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy trigger filter, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d901 for more information. Default is an empty string. Filter Search DataSet Up Vertex Entropy

exclusion	String	No	A filter that determines the alerts to exclude from Situation creation. Cookbook ignores alerts that match the exclusion filter. For details on creating a filter, see /document/preview/35090#UUIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy exclusion filter, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d901 for more information. Default is an empty string. Filter Search DataSet Up Vertex Entropy
seed_alert	String	No	A filter that determines whether to create a Situation from a seed alert. The seed alert must meet both the trigger , exclusion and seed_alert criteria to create a Situation. Cookbook considers subsequent alerts for clustering if they meet the trigger and exclusion filter criteria. Alerts that arrive prior to the seed alert that met the trigger and exclusion filter criteria do not form Situations. For details on creating a filter, see /document/preview/35090#UUIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy seed alert filter, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d901 for more information. Default is an empty string. Filter Search DataSet Up Vertex Entropy
rate	Double	No	Rate, in number of alerts per second. Cookbook clusters alerts if they arrive at a higher rate than is specified here. Cookbook uses rate together with min_sample_size and max_sample_size to determine whether to cluster alerts into Situations. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 0 which means that Cookbook does not use the rate to cluster alerts. Cookbook and Recipe Examples
min_sample_size	Positive Integer	No	Number of alerts that must arrive before the Cookbook starts to calculate the alert rate. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 5. Valid only if rate is non-zero. Cookbook and Recipe Examples
max_sample_size	Positive Integer	No	Maximum number of alerts that are considered in the alert rate calculation. When more than this number of alerts have arrived, Cookbook discards the oldest alerts and calculates the alert rate based on the number of alerts in the max_sample_size . See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 10. Valid only if rate is non-zero. Cookbook and Recipe Examples
cook_for	Positive Integer	No	<p>Minimum time period, in seconds, that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 3600 seconds (1 hour). Cookbook and Recipe Examples</p> <p>If you set a different cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for time inherit the value from the Cookbook.</p>

cook_for_extension	Positive Integer	No	<p>Time period that Cookbook can extend clustering alerts for before the Recipe resets and starts a new cluster. Setting this value enables the cook for auto-extension feature for this Cookbook. As Cookbook receives related alerts, it continues to extend the total clustering time until the max_cook_for period is reached. Used in conjunction with the max_cook_for value, the cook_for_extension period helps to ensure that Cookbook continues to cluster alerts together that are related to the same failure. The cook_for_extension period only applies to new related alerts; it does not apply to existing alerts that are updated with new events. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 0. Cookbook and Recipe Examples</p> <p>If you set a different cook_for_extension time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for_extension time inherit the value from the Cookbook.</p>
max_cook_for	Positive Integer	No	<p>Maximum time period that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. It works in conjunction with the cook_for_extension time to help ensure that Cookbook continues to cluster alerts together that are related to the same failure. This value is ignored unless the cook_for_extension time is specified. If cook_for_extension is set and this value is not set, the default is three times the cook_for value. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 0. Cookbook and Recipe Examples</p> <p>If you set a different max_cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a max_cook_for value inherit the value from the Cookbook.</p>
cluster_by	String	No	<p>Determines Cookbook's clustering behavior. Set to an empty string to use the Cookbook cluster_by setting. Set to first_match so that Cookbook adds alerts to the first cluster over the similarity threshold value. Set to closest_match to add alerts to the cluster with the highest similarity greater than the similarity threshold value. This option may be less efficient because Cookbook needs to compare alerts against each cluster in a Recipe. Default is an empty string which means the Recipe uses the Cookbook setting.</p> <p>If you set a different cluster_by value for a Recipe, it overrides the Cookbook value. Recipes without a cluster_by time inherit the value from the Cookbook.</p>
initialize_function	JSON Function Name	No	Default is initBuckets .
member_function	JSON Function	No	Default is checkBucket .

n	on Name		
can_start_cluster	JSON Function Name	No	Default is null.
use_in_recipe	JSON Function Name	No	Default is null.
similarity	Double	No	Value between 0 and 1. Default is 0.8.

Response

Endpoint **addBotRecipe** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **addBotRecipe**:

Request example

Example cURL request to create a new Bot Recipe "BotRecipe2":

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/addBotRecipe"
-H "Content-Type: application/json; charset=UTF-8" -d '{"cookbooks" :
["GrazeCookbook1"], "name": "BotRecipe2", "alert_threshold": 1}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addCookbook

A Graze API POST request that creates a new Cookbook.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addCookbook** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the Cookbook. Must be unique.
recipes	List of	Yes	A list of the Recipes in this Cookbook. You must supply at

	String		least one Recipe. If you set first_recipe_match_only to first_match , Cookbook uses the order of the Recipes in this list to determine their priority. The first Recipe has the highest priority.
description	String	No	Description of the Cookbook.
process_output_of	List of Strings	Yes	Defines the source of the alerts that Cookbook processes. You can specify one or multiple Moolets. Valid values are: Alert Workflows , AlertBuilder , AlertRulesEngine , MaintenanceWindowManager , EmptyMoolet .
run_on_startup	Boolean	No	Whether Cookbook should start when Moogfarmd starts. Default is true .
scale_by_severity	Boolean	No	Determines whether Cookbook ignores alerts with a severity of 0 (Clear). Set to true if you want Cookbook to ignore alerts with a severity of 0 (Clear). Set to false if you want Cookbook to include alerts with a severity of 0 (Clear). Default is false .
entropy_threshold	Number	No	Minimum entropy value an alert must have in order for Cookbook to consider it for clustering it into a Situation. Cookbook does not include any alerts with an entropy value below the threshold in Situations. Default is 0.0 which means that Cookbook processes all alerts.
first_recipe_match_only	Boolean	No	Defines whether Cookbook treats Recipes in priority order. If set to true , Cookbook adds an alert to a cluster created by the highest priority Recipe that meets the clustering criteria. The priority order is defined by the order of the Recipes in the recipes list. If set to false , Cookbook adds an alert to clusters in all the Recipes that meet the clustering criteria. Default is false .
cluster_by	String	No	Determines Cookbook's clustering behavior. Set to first_match so that Cookbook adds alerts to the first cluster over the similarity threshold value. Set to closest_match to add alerts to the cluster with the highest similarity greater than the similarity threshold value. This option may be less efficient because Cookbook needs to compare alerts against each cluster in a Recipe. Default is first_match . If you set a different cluster_by value for a Recipe, it overrides the Cookbook value. Recipes without a cluster_by time inherit the value from the Cookbook.
cook_for	Integer	No	Minimum time period, in seconds, that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 3600 seconds (1 hour). Cookbook and Recipe Examples If you set a different cook_for time for a Recipe, it

			overrides the Cookbook value. Recipes without a cook_for time inherit the value from the Cookbook.
cook_for_extension	Integer	No	<p>Time period that Cookbook can extend clustering alerts for before the Recipe resets and starts a new cluster. Setting this value enables the cook for auto-extension feature for this Cookbook. As Cookbook receives related alerts, it continues to extend the total clustering time until the max_cook_for period is reached. Used in conjunction with the max_cook_for value, the cook_for_extension period helps to ensure that Cookbook continues to cluster alerts together that are related to the same failure. The cook_for_extension period only applies to new related alerts; it does not apply to existing alerts that are updated with new events. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 0. Cookbook and Recipe Examples</p> <p>If you set a different cook_for_extension time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for_extension time inherit the value from the Cookbook.</p>
max_cook_for	Integer	No	<p>Maximum time period that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. It works in conjunction with the cook_for_extension time to help ensure that Cookbook continues to cluster alerts together that are related to the same failure. This value is ignored unless the cook_for_extension time is specified. If cook_for_extension is set and this value is not set, the default is three times the cook_for value. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 0. Cookbook and Recipe Examples</p> <p>If you set a different max_cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a max_cook_for value inherit the value from the Cookbook.</p>
moobot	String	No	<p>The Moobot you want Cookbook to use if there are any Bot Recipes. See /document/preview/114188#UUID1ecd7d867ea666f30327761f670221b7 for more information. Default is Cookbook.js. Recipe Types</p>

Response

Endpoint **addCookbook** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **addCookbook**:

Request example

Example cURL request to create a new Cookbook "GrazeCookBook1":

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/addCookbook"
-H "Content-Type: application/json; charset=UTF-8" -d '{"name":
"GrazeCookBook1", "process_output_of": ["Alert Workflows"], "recipes":
["Description","Source"], "run_on_startup":false,
"first_recipe_match_only":true}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addMergeGroup

A Graze API POST request that adds a new custom merge group.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addMergeGroup** takes the following request arguments:

Name	Type	Required	Description
name	String	Yes	A unique name for the custom merge group.
moolets	List	Yes	List of clustering algorithm Moolets to include in the custom merge group.
alert_threshold	Integer	No	Minimum number of alerts that must be present in a cluster before it can become a Situation. Must be greater than or equal to 1. Enter null if you want the custom merge group to use the default merge group value. Default merge group value is 2.
situation_similarity_limit	Floating Point	No	Percentage of alerts two Situations must share before they are merged for this group. Enter a value between 0 and 1. Enter null if you want the merge group to use the default merge group value.

Response

Endpoint **addMergeGroup** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **addMergeGroup**:

Request example

Example cURL request to create a new custom merge group:

```
curl -X POST
-u graze:graze
-k -v "https://example.com/graze/v1/addMergeGroup"
-H "Content-Type: application/json; charset=UTF-8"
-d '{"name":"Merge Group 1","moolets":["Time Based (Tempus)", "Recipe
2"],"alert_threshold":2,"situation_similarity_limit":0.6}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addProcess

A Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addProcess** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Process name.
description	String	No	Process description.

Response

Endpoint **addProcess** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **addProcess**:

Request example

Example cURL request to add a new process "New Proc 1" with a description:

```
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"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addService

A Graze API POST request that adds a new external service to the database. An external service is a business entity monitored by Moogsoft AIOps via event streams.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addService** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the external service you are adding.
description	String	No	Service description.

Response

Endpoint **addService** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **addService**:

Request example

Example cURL request to add service "New Service 1" with a description:

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

Response example

A successful request returns the HTTP code 200 and no response text.

addSigCorrelationInfo

A Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addSigCorrelationInfo** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

sitn_id	Number	Yes	Situation ID.
service_name	String		Name of the external service, for example, ServiceNow.
resource_id	String		ID of the external service entity to associate with this Situation.

Response

Endpoint **addSigCorrelationInfo** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **addSigCorrelationInfo**:

Request example

Example cURL request to associate resource ID "my resource 7" in service "my service 7" with Situation ID 7:

```
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"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addSituationCustomInfo

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addSituationCustomInfo** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for

			more information.
sitn_id	Number	Yes	Situation ID.
custom_info	JSON object	Yes	A JSON object containing the custom information.

Response

Endpoint **addSituationCustomInfo** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **addSituationCustomInfo**:

Request example

Example cURL request to add custom info to "field1", "field2", "field3", and "field4" in Situation ID 5:

```
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"} } }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addTempus

A Graze API POST request that adds a new Tempus Moolet.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addTempus** takes the following request arguments:

Name	Type	Requir	Description
------	------	--------	-------------

		ed	
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the Tempus algorithm. Must be unique.
description	String	No	Description of the Situations Tempus generates. Default is 'A Tempus Situation'.
process_output_of	List of Strings	Yes	Defines the source of the alerts that Tempus processes.
run_on_startup	Boolean	No	Whether Tempus should start when Moogfarmd starts. Default is true .
entropy_threshold	Number	No	Minimum entropy value for an alert to be clustered into a Situation. Tempus does not cluster any alerts with an entropy value below the threshold into Situations. The default of 0 means that all alerts are processed.
execution_interval	Number	No	Executes Tempus after a defined number of seconds. Default is 120.
window_size	Number	No	Determines the length of time, in seconds, when Tempus analyzes alerts and clusters them into a Situation each time it runs. Default window size is 1200 seconds (20 minutes). The default window size and bucket size provides 240 buckets per time period.
bucket_size	Number	No	Determines the time span, in seconds, of each bucket in which alerts are captured. Default bucket size is 5 seconds. The default window size and bucket size provides 240 buckets per time period.
arrival_spread	Number	No	Sets the acceptable latency or arrival window for each alert, in seconds. Use this to minimise or reduce the impact of multiple alerts arriving over a small amount of time and landing in separate buckets. This is a value between 1 and 60. Default is 15.
minimum_arrival_similarity	Number	No	How similar alerts must be for Tempus to consider them for clustering. Default is 0.6667.
alert_threshold	Number	No	<p>Minimum number of alerts that match the clustering criteria before the Tempus algorithm creates a Situation. Default is 4.</p> <p>When Tempus determines the number of alerts required to create a Situation, it compares the alert threshold values in Tempus and in the merge group that Tempus belongs to, and it uses the higher value. If you are using the default merge group which has an alert threshold of 2, Tempus will never create a Situation containing a single alert. If you want Cisco Crosswork Situation Manager to create Situations with a single alert, consider changing the alert threshold in the default merge group to</p>

			1 or creating custom merge groups. See /document/preview/121633#UUIDdc5f5ef14beb1148529d6f5b50806b63 for more information on updating the default merge group and setting up custom merge groups. Configure Merge Groups
partition_by	String	No	Splits clustering according to the entered component. After alerts have been clustered and before they enter merging and resolution, you can split clusters into sub-clusters based on a component of the events. For example, you can use the manager parameter to ensure that Situations only contain events from the same manager. The default of null means that no partitioning occurs. <i>Note:</i> Cisco does not recommend partitioning by components.
pre_partition	Boolean	No	Partitions event streams before clustering. You specify a component field on which the event stream will be partitioned before clustering occurs. The alerts in the resulting Situations each contain a single value for the component field chosen. The default of null means that no pre-partitioning occurs.
significance_test	String	No	Calculation that determines how significant a cluster of alerts or a potential Situation must be for Tempus to detect it. The default, Poisson1 , looks at the data of a single alert cluster to calculate how significant it is. The default is more likely to detect all significant alert clusters but with a higher risk of creating insignificant alert clusters. Use this option when your alerts originate from different networks or unrelated topologies. Poisson2 is a more thorough test that looks at an alert cluster and all alerts outside the cluster with a similar event rate. It is more likely to exclude all insignificant alert clusters but with a high risk of excluding significant alert clusters. Use this option if you expect all of your alerts to come from the same connected network. See Poisson distribution for more information.
significance_threshold	Number	No	Sets the maximum significance score for Tempus to create a Situation. The score is proportional to the probability that the alert cluster or potential Situation was coincidence. The lower the score, the more significant the cluster and the least likely it was a coincidence. This score ranges from 0 to 100. Default is 1.
detection_algorithm	String	No	Detection algorithm that Tempus uses, one of: Louvain , LouvainMulti , or SmartLocal . Default is Louvain .

Response

Endpoint **addTempus** returns the following response:

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

Code	definitions for more information.
------	---

Examples

The following examples demonstrate typical use of endpoint **addTempus**:

Request example

Example cURL request to create a new Tempus algorithm:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/addTempus" -H
"Content-Type: application/json; charset=UTF-8" -d
'{"name":"GrazeTempusTest4", "description":"Situation Generated by
Tempus","process_output_of":"Alert Workflows",
"run_on_startup":false,"entropy_threshold":0.3,"execution_interval":60,"win
dow_size":240,"bucket_size":3,"arrival_spread":9,"minimum_arrival_similarit
y":0.5,"alert_threshold":5,"partition_by":"source","significance_test":"Poi
sson2","significance_threshold":3,"detection_algorithm":"LouvainMulti"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

addThreadEntry

A Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addThreadEntry** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
thread_name	String	Yes	Name of the existing thread.
entry	String	Yes	Description of the new entry you want to create in the thread. For example, " And another thing..." . HTML and XML tags are stripped from the thread entry text. Reserved characters are converted to HTML entities, for example, & is converted to &amp; .
resolving_step	Boolean	No	Whether or not the thread entry you are adding is a resolving step.

Response

Endpoint **addThreadEntry** returns the following response:

Type	Description
------	-------------

HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.
-----------	---

Successful requests return a JSON object containing the following:

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

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **addThreadEntry**:

Request example

Example cURL request to add a new entry "Test 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}'
```

Response example

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

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

addValueRecipe

A Graze API POST request that creates a new Cookbook Recipe using Value Recipe or Value Recipe v2 recipe types. See [/document/preview/114188#UUID1ecd7d867ea666f30327761f670221b7](#) for more information. Recipe Types

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **addValueRecipe** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
cookbook	List	No	A list of the Cookbooks that this Recipe belongs to. You can add

	of Strin gs		Cookbooks here or, when you create a Cookbook, you can assign the Recipes to it.
name	Strin g	Yes	Name of the Recipe. Use a unique and descriptive name.
descriptio n	Strin g	No	Description of the Recipe. Default is the Recipe name .
version	Strin g	No	Defines whether the Recipe uses Value Recipe or Value Recipe v2. Valid values are V1 for the Value Recipe and V2 for Value Recipe v2. Default is V2. See /document/preview/114188#UUID1ecd7d867ea666f30327761f670221b7 for more information. Use addBotRecipe if you want to create a Bot Recipe.Recipe Types
alert_thre shold	Posit ive Integ er	No	Minimum number of alerts required before Cookbook creates a Situation. When Cookbook determines the number of alerts required to create a Situation, it compares the alert threshold values in the Cookbook Recipe and in the merge group that the Cookbook Recipe belongs to, and it uses the higher value. If you are using the default merge group which has an alert threshold of 2, Cookbook will never create a Situation containing a single alert. If you want Cisco Crosswork Situation Manager to create Situations with a single alert, consider changing the alert threshold in the default merge group to 1 or creating custom merge groups. See /document/preview/121633#UIDdc5f5ef14beb1148529d6f5b50806b63 for more information on updating the default merge group and setting up custom merge groups.Configure Merge Groups
trigger	Strin g	No	A filter that determines the alerts that Cookbook considers for Situation creation. Cookbook includes alerts that match the trigger filter. By default Cookbook only includes alerts with a severity of 'Critical'. For details on creating a filter, see /document/preview/35090#UIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy trigger filter, see /document/preview/114709#UID1bb978536bbf4215b156771ae826d901 for more information. Default is an empty string.Filter Search DataSet Up Vertex Entropy
exclusion	Strin g	No	A filter that determines the alerts to exclude from Situation creation. Cookbook ignores alerts that match the exclusion filter. For details on creating a filter, see /document/preview/35090#UIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy exclusion filter, see /document/preview/114709#UID1bb978536bbf4215b156771ae826d901 for more information. Default is an empty string.Filter Search DataSet Up Vertex Entropy
seed_alert	Strin g	No	A filter that determines whether to create a Situation from a seed alert. The seed alert must meet both the trigger , exclusion and seed_alert criteria to create a Situation. Cookbook considers subsequent alerts for clustering if they meet the trigger and exclusion filter criteria. Alerts that arrive prior to the seed alert that met the trigger and exclusion filter criteria do not form Situations. For details on creating a filter, see

			/document/preview/35090#UIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy seed alert filter, see /document/preview/114709#UID1bb978536bbf4215b156771ae826d901 for more information. Default is an empty string. Filter Search DataSet Up Vertex Entropy
rate	Double	No	Rate, in number of alerts per second. Cookbook clusters alerts if they arrive at a higher rate than is specified here. Cookbook uses rate together with min_sample_size and max_sample_size to determine whether to cluster alerts into Situations. See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Default is 0 which means that Cookbook does not use the rate to cluster alerts. Cookbook and Recipe Examples
min_sample_size	Positive Integer	No	Number of alerts that must arrive before the Cookbook starts to calculate the alert rate. See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Default is 5. Valid only if rate is non-zero. Cookbook and Recipe Examples
max_sample_size	Positive Integer	No	Maximum number of alerts that are considered in the alert rate calculation. When more than this number of alerts have arrived, Cookbook discards the oldest alerts and calculates the alert rate based on the number of alerts in the max_sample_size . See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Default is 10. Valid only if rate is non-zero. Cookbook and Recipe Examples
cook_for	Positive Integer	No	Minimum time period, in seconds, that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Default is 3600 seconds (1 hour). Cookbook and Recipe Examples If you set a different cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for time inherit the value from the Cookbook.
cook_for_extension	Positive Integer	No	Time period that Cookbook can extend clustering alerts for before the Recipe resets and starts a new cluster. Setting this value enables the cook for auto-extension feature for this Cookbook. As Cookbook receives related alerts, it continues to extend the total clustering time until the max_cook_for period is reached. Used in conjunction with the max_cook_for value, the cook_for_extension period helps to ensure that Cookbook continues to cluster alerts together that are related to the same failure. The cook_for_extension period only applies to new related alerts; it does not apply to existing alerts that are updated with new events. See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Default is 0. Cookbook and Recipe Examples If you set a different cook_for_extension time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for_extension time inherit the value from the Cookbook.
max_cook_for	Positive Integer	No	Maximum time period that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. It works in conjunction with the

	Integer		<p>cook_for_extension time to help ensure that Cookbook continues to cluster alerts together that are related to the same failure. This value is ignored unless the cook_for_extension time is specified. If cook_for_extension is set and this value is not set, the default is three times the cook_for value. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Default is 0. Cookbook and Recipe Examples</p> <p>If you set a different max_cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a max_cook_for value inherit the value from the Cookbook.</p>												
cluster_by	String	No	<p>Determines Cookbook's clustering behavior. Set to an empty string to use the Cookbook cluster_by setting. Set to first_match so that Cookbook adds alerts to the first cluster over the similarity threshold value. Set to closest_match to add alerts to the cluster with the highest similarity greater than the similarity threshold value. This option may be less efficient because Cookbook needs to compare alerts against each cluster in a Recipe. Default is an empty string which means the Recipe uses the Cookbook setting.</p> <p>If you set a different cluster_by value for a Recipe, it overrides the Cookbook value. Recipes without a cluster_by time inherit the value from the Cookbook.</p>												
matcher	JSON Structure	Yes	<p>A JSON structure containing:</p> <p>hop_limit: Maximum number of hops between the alert source nodes in order for the alerts to qualify for clustering. Cisco Crosswork Situation Manager measures hop limit from the first alert that formed the Situation and always follows the shortest possible route in the network. A hop is the jump between two directly connected nodes in a network. For more information on hops, see /document/preview/113737#UUID6cc2007bfdb8ce5fe9adef8dbcb8307d. To set a hop limit, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d901 for more information. Vertex Entropy Set Up Vertex Entropy</p> <p>You can only use a hop limit if you have imported your network topology into the system. See /document/preview/11702#UUID517934cfa987aa330c16e11e1be74be8 for details. Import a Topology</p> <p>components: Values that alerts must match for Cookbook to include them in a Situation. You can provide multiple values such as source, description, service or custom info fields. An array of JSON objects, each containing:</p> <table> <thead> <tr> <th>Name</th><th>Type</th><th>Required</th><th>Description</th></tr> </thead> <tbody> <tr> <td>name</td><td>String</td><td>Yes</td><td>Name of the component.</td></tr> <tr> <td>similarity</td><td>Double</td><td>Yes</td><td>Similarity threshold that the component must meet for Cookbook to cluster the alert into a Situation.</td></tr> </tbody> </table>	Name	Type	Required	Description	name	String	Yes	Name of the component.	similarity	Double	Yes	Similarity threshold that the component must meet for Cookbook to cluster the alert into a Situation.
Name	Type	Required	Description												
name	String	Yes	Name of the component.												
similarity	Double	Yes	Similarity threshold that the component must meet for Cookbook to cluster the alert into a Situation.												

			shingle_size	Integer	No	Shingle size for Cookbook to use to determine the similarity between different strings. The shingle size is only valid for Recipe Value v2 recipes. Default is -1 which means that Cookbook uses words to determine similarity. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for more details. Recipe Types
			treat_as	String	No	Determines whether Cookbook treats the component as a string or matches each value in the list individually. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for details. Valid values are List and String . Default is String . Recipe Types
			case_sensitive	Boolean	No	Enables or disables case sensitive when comparing strings. Case sensitivity is only valid for Recipe Value recipes. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for more details. Default is true which means that strings are treated as case sensitive. Recipe Types

Response

Endpoint **addValueRecipe** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **addValueRecipe**:

Request example

Example cURL request to add a new Recipe "GrazeRecipe2":

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/addValueRecipe" -H "Content-Type:
application/json; charset=UTF-8" -d '{"cookbook" : "GrazeCook1", "name":
"GrazeRecipe2", "alert_threshold" : 1,
"matcher" : { "hop_limit" : 0},
"components" : [{ "name": "custom_1",
"similarity": 0.2,
"shingle_size": 2 } ]
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

applyNewLicense

A Graze API POST request that adds a Cisco Crosswork Situation Manager license via Graze.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **applyNewLicense** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
license	String	Yes	A valid license key.

Response

Endpoint **applyNewLicense** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **applyNewLicense**:

Request example

Example cURL request to add a valid license:

```
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>"}
```

Response example

A successful request returns the HTTP code 200 and no response text.

assignAlert

A Graze API POST request that assigns the specified user as the owner of the specified alert ID.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **assignAlert** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	Alert ID.
user_id	Number	No, if you specify username .	ID of the user to be assigned as the owner of the alert. You must provide the user_id or the username .

username	String	No, if you specify user_id .	Username of the user to be assigned as the owner of the alert. You must provide the user_id or the username .
-----------------	--------	-------------------------------------	---

Response

Endpoint **assignAlert** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	No
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **assignAlert**:

Request example

Example cURL request to username "network1" to alert ID 7:

```
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" }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

assignAndAcknowledgeAlert

A Graze API POST request that assigns and acknowledges the specified user as the owner of the specified alert ID.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **assignAndAcknowledgeAlert** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	Alert ID.

user_id	Number	No, if you specify username .	ID of the user to be assigned as the owner of the alert. You must provide the user_id or the username .
username	String	No, if you specify user_id .	Username of the user to be assigned as the owner of the alert. You must provide the user_id or the username .

Response

Endpoint **assignAndAcknowledgeAlert** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	No
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **assignAndAcknowledgeAlert**:

Request example

Example cURL request to assign user "Cloud DevOps 1" as the owner of alert 432:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/assignAndAcknowledgeAlert" -H "Content-Type:
application/json; charset=UTF-8" -d '{"alert_id" : 432, "username" : "Cloud
DevOps 1" }'???
```

Response example

A successful request returns the HTTP code 200 and no response text.

assignAndAcknowledgeSituation

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **assignAndAcknowledgeSituation** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request.

			See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
user_id	Number	Yes	ID of the user to be assigned as the owner of the Situation.
username	String	No	Username of the user to be assigned to the Situation.

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

Response

Endpoint **assignAndAcknowledgeSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **assignAndAcknowledgeSituation**:

Request example

Example cURL request:

Response example

A successful request returns the HTTP code 200 and no response text.

assignSituation

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **assignSituation** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

sitn_id	Number	Yes	Situation ID.
user_id	Number	Yes	User ID.
username	String	No	A valid username.

Response

Endpoint **assignSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **assignSituation**:

Request example

Example cURL request:

```
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 }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

assignTeamsToSituation

A Graze API 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 Moollet can no longer modify its team assignment. See [Teams Manager Moollet](#) for more information. Teams Manager Moollet

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **assignTeamsToSituation** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
team_ids	List	No, if team_names is used.	A list of team IDs to assign to the Situation. Specify an empty list to unassign all teams from the Situation.
team_names	List	No, if team_ids is used.	A list of team names to assign to the Situation. Specify an empty list to unassign all teams from the Situation.

Response

Endpoint **assignTeamsToSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing one of the following, depending on the request argument used:

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.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **assignTeamsToSituation**:

Example assigning team IDs

Example cURL request to assign team IDs 1 and 2 to Situation 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
]}'
```

Example response returning the team IDs (1 and 2) that have been successfully assigned to the Situation:

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

Example assigning team names

Example cURL request to assign teams "Network_US" and "Network_UK" to Situation 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" : [
"Network_US", "Network_UK" ]}'
```

Example response returning the team names ("Network_US" and "Network_UK") that have been successfully assigned to the Situation:

```
{"team_names" : [ "Network_US", "Network_UK" ]}
```

Example unassigning teams

Example cURL request to unassign all teams from Situation 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" : []}'
```

Example response returning an empty list showing that all the teams have been successfully unassigned from the Situation:

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

authenticate

A Graze API 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.

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**.

If you make regular Graze requests within a one hour timeframe, you are considered active and your session does not expire. Inactive sessions are logged out after one hour, and you must make a new **authenticate** request to get a new valid **auth_token**.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **authenticate** takes the following request arguments:

Name	Type	Required	Description
username	String	Yes	A valid Cisco Crosswork Situation Manager username.
password	String	Yes	The username's corresponding password.

Response

Endpoint **authenticate** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

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

Examples

The following examples demonstrate typical use of endpoint **authenticate**:

Request examples

Example cURL request to return an authorization token for username "JohnJones" and password "password":

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

Example cURL request to return an authorization token for the Graze user:

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

Response example

Example response returning the authorization token:

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

checkSituationFlag

A Graze API GET request that checks whether a flag is associated with a Situation.

See [Situation Flags](#) for more information on Cisco Crosswork Situation Manager Situation flags.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **checkSituationFlag** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	ID of the Situation to be checked.
flag	Number	Yes	Flag to be checked for the specified Situation ID.

Response

Endpoint **checkSituationFlag** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Type	Description
Boolean	Whether or not the flag is associated with the specified Situation.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **checkSituationFlag**:

Request example

Example cURL request to request a Boolean whether the specified Situation has the flag "NOTIFIED" associated with it.

```
curl -X GET -u graze:graze -k -v
https://localhost/graze/v1/checkSituationFlag?sitn_id=1&flag=NOTIFIED
```

Response example

Example response returning the Boolean value **true** because the Situation contains the specified flag:

```
true
```

closeAlert

A Graze API POST request that closes one or more alerts.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **closeAlert** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	No, if alert_ids is used.	A single alert ID. You must provide a single alert_id or a list of alert_ids .

alert_ids	Number list	No, if alert_id is used.	A list of alert IDs. You must provide a single alert_id or a list of alert_ids .
thread_entry_comment	String	No	Thread entry comment you want to add to the closed alert. HTML and XML tags are stripped from the thread entry text. Reserved characters are converted to HTML entities, for example, & is converted to &amp; .

Response

Endpoint **closeAlert** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **closeAlert**:

Request example

Example cURL request to close alert IDs 78, 234, and 737:

```
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" }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

closeSituation

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **closeSituation** takes the following request arguments:

Cisco Systems, Inc. www.cisco.com

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
resolution	Number	Yes	Determines what to do with the alerts in the Situation: 0 = Close no alerts. 1 = Close all alerts in this Situation. 2 = Close only alerts unique to this Situation.

Response

Endpoint **closeSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **closeSituation**:

Request example

Example cURL request to close Situation 7 and leave all its alerts open:

```
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 }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

createMaintenanceWindow

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **createMaintenanceWindow** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the maintenance window.
description	String	Yes	Description of the maintenance window.
filter	String	Yes	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. If the filter includes a backslash in the filter string, you need to double escape these to maintain the backslash character in the filter string. However, if you have \t , \n , \b , or \r followed by a backslash in the path, you do not need to pass any extra backslashes. See the Example of escaped characters .
start_date_time	Number	Yes	Start time of the maintenance window. This must be in Unix epoch time and may be up to 5 years in the future.
duration	Number	Yes	Duration of the maintenance window in seconds. The minimum duration is 1 second and the maximum is 157784630 seconds (5 years).
forward_alerts	Boolean	Yes	Whether or not alerts should be forwarded to the next Moolet in the processing chain.
recurring_period	Number	No	Whether or not this is a recurring maintenance window. Set this to: 1 for a recurring maintenance window. 0 for a one-time maintenance window.If not specified, default is 0 . If you set this property 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. Valid values are: 2 = daily 3 = weekly 4 = monthlyDefault is 0 if recurring_period is set to 0 .
timezone	String	No	Time zone that you want the maintenance window to be created in. Default is the time zone of the user that makes the request. If the user has a "SYSTEM" time zone, Cisco Crosswork Situation Manager uses the MoogSvr time zone. The time zone must be a valid entry in the IANA Time Zone Database . When scheduling recurring maintenance windows, Cisco Crosswork Situation Manager takes into account any daylight savings time changes for

			the time zone.
--	--	--	----------------

Response

Endpoint **createMaintenanceWindow** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
window_id	Number	ID of the new maintenance window.

Examples

The following examples demonstrate typical use of endpoint **createMaintenanceWindow**:

Request examples

Example cURL request to create a window, which recurs once a month (from its start_date_time), with a filter where the source 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 the source is equal to "hostIsDown":

```
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": "hostIsDown", "type": "LEAF" }, "start_date_time":
1473849237, "duration": 55800, "forward_alerts": false}'
```

Example cURL request to create a daily maintenance window in the "America/New York" time zone:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/createMaintenanceWindow" -H "Content-Type:
application/json; charset=UTF-8" -d '{"name": "window",
"description": "window with specified timezone", "filter": "source =
\"server1\" and external_id in (\"value1\", \"value2\", \"value3\")",
"start_date_time": 1564566188, "duration": 3600, "forward_alerts": false,
"recurring_period": 1, "recurring_period_units": 2, "timezone":
"America/New_York"}'
```

Example of escaped characters

Example cURL request using multiple escaped backslash characters in the filter to maintain the correct characters in the filter string:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/createMaintenanceWindow" -H "plication/json;
charset=UTF-8" -d '{"name":"LON Server 1", "description":"[10453] ",
"filter":"NOT (alert_id IS NULL) AND (agent_location MATCHES \"LON_S1\")
AND (custom_info.eventDetails.alert_customer MATCHES \"LON\") AND (manager
MATCHES \"LOGFILE\") AND (class MATCHES \"could not execute backup\") AND
(custom_info.eventDetails.field_1 MATCHES
\"C:\\\\\\\\\\\\\\\\\\\\LON_SVR1\\\\Backup\\\\2019\\\\)\", "start_date_time":1480483478,
"duration":86400, "recurring_period_units":2, "recurring_period":1,
"forward_alerts":false}'
```

The endpoint **createMaintenanceWindow** creates the correct filter:

```
((((( NOT (alert_id IS NULL)) AND (`Agent location` MATCHES "LON_S1")) AND
(custom_info.eventDetails.alert_customer MATCHES "LON")) AND (Manager
MATCHES "LOGFILE")) AND (Class MATCHES "could not execute backup")) AND
(custom_info.eventDetails.field_1 MATCHES "C:\\\\\\\\\\\\LON_SVR1\\Backup\\2019")
```

Response example

A successful request returns the ID of the new maintenance window:

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

createSecurityRealm

A Graze API 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 are logged into Cisco Crosswork Situation Manager using the default realm before using this request. Cisco Crosswork Situation Manager may log out users when the new realm becomes active.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **createSecurityRealm** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the security realm.
type	String	Yes	Security realm type. This must be "SAML2" .
active	Boolean	Yes	Determines whether the new realm is active in Cisco Crosswork Situation Manager on creation. 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	Yes	JSON object containing the realm configuration. For information on the configuration properties, see /document/preview/11687#UUID34c1daabda9eb021bd2a4ba43eda6

			83b.Security Configuration Reference Upload your IdP metadata file using idpMetadata or specify the location of the file using idpMetadataUrl. For example: <pre>"idpMetadataUrl":"http://&lt;location_of_idp_metadata&gt;";</pre> <pre>"idpMetadata":"&lt;raw_ipd_metadata.xml&gt;";</pre>
--	--	--	---

Response

Endpoint **createSecurityRealm** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **createSecurityRealm**:

Request example

Example cURL request to generate a security realm:

```
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
}
}
```

Response example

A successful request returns the HTTP code 200 and no response text.

createSituation

A Graze API 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:

1. Status: Opened

2. Moderator: none assigned

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **createSituation** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
description	String	Yes	Description of the new Situation.

Response

Endpoint **createSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
sitn_id	Number	ID of the newly created Situation.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	No
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **createSituation**:

Request example

Example cURL request to create a Situation with the description "Database Outage 08/06/2019":

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/createSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"description" : "Database Outage
08/06/2019"}'
```

Response example

Example response returning the ID of the newly created Situation:

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

createTeam

A Graze API POST request that creates a new team.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **createTeam** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	New team name. Must be unique.
alert_filter	String	Yes	Team alerts filter. Either a SQL like filter or an JSON representation of the filter.
services	JSON List	Yes	List of the team services or IDs.
sig_filter	String	Yes	Situation filters. Either a SQL like filter or an JSON representation of the filter.
landing_page	String	Yes	Team default landing page.
active	Boolean	Yes	False if the team is inactive, true if the team is active. Default is true .
description	String	Yes	Team description.
users	List of numbers or strings	Yes	Team users (either IDs or usernames).

Response

Endpoint **createTeam** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
team_id	Number	ID of the new team.

Examples

The following examples demonstrate typical use of endpoint **createTeam**:

Request example

Example cURL request to create a team called "my team name" consisting of user1, user2, and user3:

```
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"]}'
```

Response example

Example response returning the ID of the created team:

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

createThread

A Graze API POST request that creates a new thread for a specified Situation. Threads are comments or 'story activity' on Situations.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **createThread** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	ID of the Situation you want to create a new thread for.
thread_name	String	Yes	Name of the new thread.

Response

Endpoint **createThread** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **createThread**:

Request example

Example cURL request to create a new thread "Thread 0958" on Situation ID 176:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/createThread"
-H "Content-Type: application/json; charset=UTF-8" -d '{"sitn_id" : 176,
"thread_name" : "Thread 0958"}
```

Response example

A successful request returns the HTTP code 200 and no response text.

createThreadEntry

Note: This endpoint has been superseded. Use [addThreadEntry](#) instead. All new functionality will be delivered in `addThreadEntry`.

A Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **createThreadEntry** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
thread_name	String	Yes	Name of the existing thread.
entry	String	Yes	Description of the new entry you want to create in the thread. For example, "And another thing...". HTML and XML tags are stripped from the thread entry text. Reserved characters are converted to HTML entities, for example, & is converted to &amp; .
resolving_step	Boolean	No	Whether or not the thread entry you are creating is a resolving step.

Response

Endpoint **createThreadEntry** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

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

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **createThreadEntry**:

Request 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}'
```

Response example

Example response showing that the new thread entry was successfully created::

```
true
```

createUser

A Graze API POST request that creates a new user.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **createUser** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
username	String	Yes	Login username for the new user. Must be unique.
password	String	Yes	New user password, only valid for DB realm.
active	Boolean	Yes	true if the user is active, false if the user is inactive. Defaults to true .
email	String	Yes	User's email address.
fullname	String	Yes	User's full name.
roles	JSON list	Yes	List of either the role IDs or the role names. For

			example, "roles":["Super User"] .
primary_group	String or Number	Yes	User's primary group name or primary group ID.
department	String or Number	Yes	User's department ID or department name.
joined	Number	Yes	Time the user joined in Unix epoch time.
timezone	String	Yes	User's timezone.
contact_num	String	Yes	User's phone number.
session_expiry	Number	Yes	Number of minutes after which the user's session expires. Defaults to the system default.
teams	JSON list of Numbers or Strings	Yes	List of the user's team names or team IDs.

Response

Endpoint **createUser** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
user_id	Number	ID of the new user.

Examples

The following examples demonstrate typical use of endpoint **createUser**:

Request example

Example cURL request to create a new user "johndoe1":

```
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, "teams": ["my team 1", "my team 2", "my team 3"], "properties": null}'
```

Response example

Example response returning the new user ID:

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

createWorkflow

A Graze API POST request that creates a new workflow in the Workflow Engine. The new workflow is automatically active.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **createWorkflow** takes the following request arguments:

Name	Type	Required	Description																
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.																
moolet_name	String	Yes	Name of the Workflow Engine Moolet that the new workflow belongs to.																
workflow_name	String	Yes	Name of the new workflow. Must be unique.																
description	String	No	Description of the new workflow.																
entry_filter	JSON	No	Filter to determine which events, alerts or Situations can enter the workflow. You can use an SQL-like query. See /document/preview/124174#UIDbd640e729795a2b9ead884be59a6e23f for more information on creating SQL-like filters. Leave empty for the workflow to accept all events, alerts or Situations. Filter Search Data																
sweep_up_filter	JSON	No	Filter to intake any additional events, alerts or Situations from the database. You can use an SQL-like query. See /document/preview/124174#UIDbd640e729795a2b9ead884be59a6e23f for more information on creating SQL-like filters. Filter Search Data																
first_match_only	Boolean	Yes	If enabled, events, alerts, and Situations only pass through actions on the first time they enter this workflow.																
operations	JSON List	Yes	<p>List of properties relating to each operation:</p> <table> <tr> <th>Name</th><th>Type</th><th>Required</th><th>Description</th></tr> <tr> <td>type</td><td>String</td><td>Yes</td><td>Type of operation. Options are: 'action', 'decision' and 'delay'.</td></tr> <tr> <td>operation_name</td><td>String</td><td>Yes, for 'action' and 'decision' types.</td><td>Name of the operation.</td></tr> <tr> <td>function_name</td><td>String</td><td>Yes, for 'action' and</td><td>Name of the function.</td></tr> </table>	Name	Type	Required	Description	type	String	Yes	Type of operation. Options are: 'action', 'decision' and 'delay'.	operation_name	String	Yes, for 'action' and 'decision' types.	Name of the operation.	function_name	String	Yes, for 'action' and	Name of the function.
Name	Type	Required	Description																
type	String	Yes	Type of operation. Options are: 'action', 'decision' and 'delay'.																
operation_name	String	Yes, for 'action' and 'decision' types.	Name of the operation.																
function_name	String	Yes, for 'action' and	Name of the function.																

						'decision' types.
			forwarding_behavior	String	No	Forwarding behavior for the function. One of: always forward : The function always forwards the object to the next workflow. stop this workflow : The function stops this workflow and the object moves to the next workflow. stop all workflows : The function stops all workflows for this object. Default is always forward . Only valid for 'action' and 'decision' types.
			function_args	JSON Object	No	Arguments for the function.
			duration	Integer	Yes, for 'delay' type.	Length of time before the message goes to the next operation.
			reset	Boolean	Yes, for 'delay' type.	Determines whether the timer resets after each occurrence. Not available if you have set a workflow to first_match_only . The timer is reset only if an occurrence with the same ID is received (alert_id or situation_id) within the current 'delay' timeframe.

Response

Endpoint **createWorkflow** returns the following response:

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

Code	definitions for more information.
------	---

Successful requests return a JSON object containing the following:

Type	Description
JSON Object	A JSON object containing the ID of the newly created workflow.

Examples

The following examples demonstrate typical use of endpoint **createWorkflow**:

Request example

Example cURL request to create a new workflow with a [/document/preview/120685#UIDa4320c8e3b5b679f9b22e4aefc7222b5](#) action:setCustomInfoValue

```
curl -X POST -u graze:graze -k \
-v "https://localhost/graze/v1/createWorkflow" \
-H "Content-Type: application/json; charset=UTF-8" \
--data ' {
  "first_match_only": false,
  "operations": [
    {
      "duration": 0,
      "reset": false,
      "type": "delay"
    },
    {
      "operation_name": "set support team value in custom info",
      "function_name": "setCustomInfoValue",
      "forwarding_behavior": "always forward",
      "function_args": {
        "value": "NOC",
        "key": "support_team"
      },
      "type": "action"
    }
  ],
  "moolet_name": "Alert Workflows",
  "workflow_name": "Alert Workflow Example",
  "entry_filter": {},
  "active": true,
  "description": "Alert Workflow API Example",
  "sweep_up_filter": {}
}'
```

Response example

Example response returning the new workflow ID:

```
{"id":12}
```

deassignAlert

A Graze API POST request that deassigns the current owner from the specified alert.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deassignAlert** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	ID of the alert that you want to deassign the owner from.

Response

Endpoint **deassignAlert** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	No
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **deassignAlert**:

Request example

Example cURL request to deassign the current owner from alert ID 7:

```
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}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deassignSituation

A Graze API POST request that deassigns the current moderator from the Situation for a specified Situation ID.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deassignSituation** takes the following request arguments:

Name	Type	Required	Description
------	------	----------	-------------

auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.

Response

Endpoint **deassignSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **deassignSituation**:

Request example

Example cURL request to deassign the current moderator from Situation 7:

```
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}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deleteCookbook

A Graze API POST request that deletes an existing Cookbook.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deleteCookbook** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the Cookbook that you want to delete.

Response

Endpoint **deleteCookbook** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **deleteCookbook**:

Request example

Example cURL request to delete Cookbook "GrazeCookBook1":

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/deleteCookbook" -H "Content-Type:
application/json; charset=UTF-8" -d '{"name" : "GrazeCookBook1"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deleteMaintenanceWindow

A Graze API POST request that deletes a single maintenance window.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deleteMaintenanceWindow** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
id	Number	Yes	ID of the maintenance window you want to delete.

Response

Endpoint **deleteMaintenanceWindow** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **deleteMaintenanceWindow**:

Request 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}]'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deleteMaintenanceWindows

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deleteMaintenanceWindows** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
filter	String	Yes	SQL or JSON filter to match maintenance windows that you want to delete.

Response

Endpoint **deleteMaintenanceWindows** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **deleteMaintenanceWindows**:

Request examples

Example cURL request to delete maintenance windows that match the filter where the maintenance window 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)}'curl -X POST
-u graze:graze -k -v "https://localhost/graze/v1/deleteMaintenanceWindow" -
H "Content-Type: application/json; charset=UTF-8" -d '{"id"[123]}'
```

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\""}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deleteMergeGroup

A Graze API POST request that deletes an existing custom merge group.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deleteMergeGroup** takes the following request arguments:

Name	Type	Required	Description
name	String	Yes	Name of the merge group to delete.

Response

Endpoint **deleteMergeGroup** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **deleteMergeGroup**:

Request example

Example cURL request to delete the custom merge group "Merge Group 1":

```
curl -X POST -u graze:graze -k
"https://localhost/graze/v1/deleteMergeGroup"
-H "Content-Type: application/json; charset=UTF-8"
--data '{
    "name" : "Merge Group 1"
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deleteRecipe

A Graze API POST request that deletes an existing Cookbook Recipe. You can use this endpoint to delete Recipes of all recipe types: Value Recipe, Value Recipe V2, and Bot Recipe.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deleteRecipe** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the Cookbook Recipe that you want to delete.

Response

Endpoint **deleteRecipe** returns the following response:

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

Code	definitions for more information.
------	---

Examples

The following examples demonstrate typical use of endpoint **deleteRecipe**:

Request example

Example cURL request to delete Recipe "GrazeRecipe1":

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/deleteRecipe"
-H "Content-Type: application/json; charset=UTF-8" -d '{"name" :
"GrazeRecipe1"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deleteTeam

A Graze API POST request that deletes a single team.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deleteTeam** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	No	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
team_id	String	Yes	ID of the team you want to delete.

Response

Endpoint **deleteTeam** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **deleteTeam**:

Request example

Example cURL request to delete team ID 33.

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1/deleteTeam" -
-data-urlencode 'team_id=33'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deleteTempus

A Graze API POST request that deletes an existing Tempus Moollet.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deleteTempus** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the Tempus Moolet you want to delete.

Response

Endpoint **deleteTempus** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **deleteTempus**:

Request example

Example cURL request to delete Tempus algorithm "newTempus":

```
curl -X POST -u graze:graze -k "https://localhost/graze/v1/deleteTempus" -H
"Content-Type: application/json; charset=UTF-8" --data '{ "name" :
"newTempus" }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

deleteWorkflow

A Graze API POST request that deletes a workflow from the Workflow Engine.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **deleteWorkflow** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
id	Integer	Yes	ID of the workflow you want to delete.

Response

Endpoint **deleteWorkflow** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **deleteWorkflow**:

Request example

Example cURL request to delete workflow ID 12:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/deleteWorkflow" -H "Content-Type:
application/json; charset=UTF-8" --data '{"id":12}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

findMaintenanceWindows

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **findMaintenanceWindows** returns the following response:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
filter	String	Yes	SQL or JSON filter to match maintenance windows that you want to find.
limit	Number	No	Maximum number of windows to fetch. Defaults to 100.

Response

Endpoint **findMaintenanceWindows** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
windows	Object	A list of objects containing details of the maintenance windows that match the filter.

Examples

The following examples demonstrate typical use of endpoint **findMaintenanceWindows**:

Request example

Example cURL request to return the first two maintenance windows that match the filter where the description is "MyMaintenanceWindow":

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

Response example

A successful request returns:

```
{
  "windows": [
    {
      "del_flag": false,
      "forward_alerts": false,
      "last_updated": 1567721780,
      "timezone": "Europe/London",
      "description": "Maintenance of server LON_DB_375",
      "recurring_period_units": 4,
      "filter": "{ \"column\": \"source\", \"op\": 10, \"value\": \"LON_DB_375\", \"type\": \"LEAF\" }",
      "duration": 3600,
      "recurring_period": 1,
      "name": "Server Maintenance",
      "updated_by": 3,
      "id": 2,
      "start_date_time": 1567812600
    },
    {
      "del_flag": false,
      "forward_alerts": true,
      "last_updated": 1567611914,
      "timezone": "Europe/London",
      "description": "Server maintenance scheduled every weekend",
      "recurring_period_units": 3,
      "filter": "{ \"column\": \"description\", \"op\": 10, \"value\": \"maintenance\", \"type\": \"LEAF\" }",
      "duration": 7200,
      "recurring_period": 1,
      "name": "Service Maintenance Window",
      "updated_by": 3,
      "id": 1,
      "start_date_time": 1567897200
    }
  ]
}
```

getActiveSituationIds

A Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getActiveSituationIds** takes the following request argument:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

There are no other arguments because this endpoint returns data on all active Situationses.

Response

Endpoint **getActiveSituationIds** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
total_situations	Number	Total number of active Situations.
sitn_ids	List	A list of active Situation IDs.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getActiveSituationIds**:

Request example

Example cURL request to return all active Situations in Cisco Crosswork Situation Manager:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/getActiveSituationIds"
```

Response example

Example response returning the IDs of ten Situations:

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

getAlertActions

A Graze API GET request that returns the actions for one or more 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getAlertActions** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_ids	Number	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 of the period you want to retrieve alert actions for. This is in Unix epoch time in seconds.
to	Number	No	End time of the period you want to retrieve alert actions for. This is in Unix epoch time in seconds.

Response

Endpoint **getAlertActions** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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	Timestamp of the action.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes

Closed Situation in historic database	Yes
---------------------------------------	-----

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getAlertActions**:

Request Examples

Example cURL request 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://docsdev.moog.cloud/graze/v1/getAlertActions" --data-urlencode
'alert_ids=[1, 2, 3, 6]' --data-urlencode 'actions=[8, 9]' --data-urlencode
'limit=50' --data-urlencode 'start=0'
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 request 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'
```

Response example

Example response returning the actions for alert ID 313:

```
[{
  "uid": 49,
  "action_code": 9,
  "description": "Alert Closed",
  "details": {},
  "alert_id": 313,
  "timed_at": 1557504912
},{
  "uid": 49,
  "action_code": 8,
  "description": "Alert Resolved",
  "details": {},
  "alert_id": 313,
  "timed_at": 1557504393
},{
  "uid": 3,
  "action_code": 10,
  "description": "Ran Tool",
  "details": {
    "tool_id": 271,
    "tool": "get data"
  },
  "alert_id": 313,
  "type": "event",
  "timed_at": 1557321088,
```

```
"username": "admin"
}
```

getAlertDetails

A Graze API GET request that returns details, such as the description or severity, of a specified alert.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getAlertDetails** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	Alert ID.

Response

Endpoint **getAlertDetails** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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	Agent name associated with this alert. *
agent_location	String	Agent location associated with this alert. *
alert_id	Number	Alert ID.
class	String	Class associated with this alert. *
count	Number	Number of times that this alert has occurred.
custom_info	JSON object	A JSON object containing the custom information.
description	String	Description associated with this alert. *
entropy	Number	Entropy value of the alert, the measure of probability that an alert will arrive in the system at any given time. This is a value between 0 (very certain) and 1 (very uncertain).
external_id	String	External ID associated with this alert. *
first_event_time	Number	Timestamp (in Unix epoch time) of the first occurrence of the alert.
int_last_event_time	Number	Internal Cisco Crosswork Situation Manager timestamp (in Unix epoch time) of the last occurrence of this alert.

last_event_time	Number	Timestamp (in Unix epoch time) of the last occurrence of this alert.
last_state_change	Number	Timestamp (in Unix epoch time) of the last state change of this alert.
manager	String	Manager name associated with this alert. *
owner	Number	ID of the user that this alert is assigned to.
severity	Number	The severity of the alert as an integer: 0 = Clear 1 = Indeterminate 2 = Warning 3 = Minor 4 = Major 5 = Critical
signature	String	Unique alert identifier.
significance	Number	Significance of the alert as an integer: 0 = Collateral 1 = Related 2 = Impacting 3 = Causal
source	String	Source associated with this alert. *
source_id	String	Source ID associated with the alert. *
state	Number	Indicates the lifecycle state of the alert.
type	String	Type associated with this alert. *

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

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getAlertDetails**:

Request example

Example cURL request to return the details for alert ID 3968:

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

Response example

Example response returning the details of alert ID 3968:

```
{
  "active_sitn_list":[1],
  "agent":"TestBed",
  "agent_location":"localhost",
  "alert_id":3968,
  "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 Graze API GET request that returns the total number of alerts, and a list of the alert IDs, for a specified alert filter and a limit.

Note: Take special care when using endpoint **getAlertIds**. Overuse of this endpoint can have a negative impact on the backend datastore.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getAlertIds** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
query	String		A JSON or SQL-like alert filter.

limit	Number		Maximum number of alert IDs to return.
--------------	--------	--	--

Response

Endpoint **getAlertIds** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
total_alerts	Number	Total number of alerts, or unique alerts.
alert_ids	Number list	A list of alert IDs.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getAlertIds**:

Request example

Example cURL request to return the first 20 alert IDs that satisfy the filter where the agent is not SYSLOG and the 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'
```

Response example

Example response returning the first 20 alert IDs:

```
{"total_alerts":20,"alert_ids":[78,234,737,1253,1459,1733,2166,2653,2855,3133,3414,3538,3729,3905,3991,4110,4160,4536,4692,4701]}
```

getCookbooks

A Graze API GET request that returns all the Cookbooks in Cisco Crosswork Situation Manager.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getCookbooks** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

There are no other arguments because this endpoint returns data on all Cookbooks in Cisco Crosswork Situation Manager.

Response

Endpoint **getCookbooks** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Type	Description
List of JSON Objects	A list of all the Cookbooks in Cisco Crosswork Situation Manager and all their details.

Examples

The following examples demonstrate typical use of endpoint **getCookbooks**:

Request example

Example cURL request to return all the Cookbooks in Cisco Crosswork Situation Manager:

```
curl -X GET -u graze:graze -k -v "https://localhost/graze/v1/getCookbooks"
```

Response example

Example response returning the details of the "Default Cookbook":

```
[
  {
    "recipes": [
      "Description",
      "Source",
      "Criticals"
    ],
    "run_on_startup": null,
    "description": "",
    "moobot": "Cookbook.js",
    "cluster_by": null,
    "cook_for": 900,
    "first_recipe_match_only": true,
    "max_cook_for": null,
    "cook_for_extension": null,
    "name": "Default Cookbook",
    "entropy_threshold": 0,
    "process_output_of": "Alert Workflows",
    "scale_by_severity": false
  }
]
```

```
}
]
```

getDefaultMergeGroup

A Graze API GET request that returns details of the default merge group in Cisco Crosswork Situation Manager.

Clustering algorithms, such as Cookbook and Tempus, use the default values in the default merge group unless you have set up custom merge groups with different values to merge Situations from these clustering algorithms. You can set up merge groups using the UI (see </document/preview/121633#UUIDdc5f5ef14beb1148529d6f5b50806b63> for details) or using the Graze API endpoint [addMergeGroup](#). Configure Merge Groups

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getDefaultMergeGroup** does not take any request arguments.

Response

Endpoint **getDefaultMergeGroup** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
alert_threshold	Integer	Minimum number of alerts that must be present in a cluster before it can become a Situation in the merge group. Default value is 2.
situation_similarity_limit	Floating Point	Percentage of alerts two Situations must share before they are merged for this group. A value between 0 and 1. Default value is 0.7.

Examples

The following examples demonstrate typical use of endpoint **getDefaultMergeGroup**:

Request example

Example cURL request to return the default merge group in Cisco Crosswork Situation Manager:

```
curl -G
-u graze:graze
-k "https://example.com/graze/v1/getDefaultMergeGroup"
```

Response example

Example response returning details of the default merge group in Cisco Crosswork Situation Manager:

```
{
  "alert_threshold": 2,
  "situation_similarity_limit": 0.7
}
```

getIntegrationConfig

A Graze API 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
```

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getIntegrationConfig** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
integration_id	Number	Yes	A unique identifier given to each integration by Cisco Crosswork Situation Manager.

Response

Endpoint **getIntegrationConfig** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
config	Object	An object containing the integration configuration details.

Examples

The following examples demonstrate typical use of endpoint **getIntegrationConfig**:

Request example

Example cURL request to return the configuration for the integration with ID 1:

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

Response example

Example return of a JSON object containing the integration configuration details:

```
{
  "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 Graze API GET request that returns maintenance windows based on how many should be returned. Only returns active recurring windows and scheduled maintenance windows, not expired or deleted maintenance windows.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getMaintenanceWindows** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
start	Number	Yes	Number of the first maintenance window to return.
limit	Number	Yes	Maximum number of maintenance windows to return.

Response

Endpoint **getMaintenanceWindows** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
windows	Array	An array of objects containing the details of the returned maintenance windows.

Examples

The following examples demonstrate typical use of endpoint **getMaintenanceWindows**:

Request example

Example cURL request 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'
```

Response example

Example successful response returning details of two maintenance windows:

```
{
  "windows": [
    {
      "filter": "{ \"column\": \"type\", \"op\": 10, \"value\": \"FireInServerRoom\", \"type\": \"LEAF\" }",
      "duration": 3600,
      "recurring_period": 1,
      "del_flag": false,
      "forward_alerts": false,
      "last_updated": 1499425460,
      "timezone": "Europe/London",
      "name": "My window 1",
      "updated_by": 5,
      "description": "My description 1",
      "id": 1,
      "recurring_period_units": 3,
      "start_date_time": 1499425427,
      "timezone": "America/New_York"
    },
    {
      "filter": "{ \"column\": \"description\", \"op\": 10, \"value\":
```

```

\"SVR_LON_3451\", \"type\": \"LEAF\" },
    \"duration\":3600,
    \"recurring_period\":0,
    \"del_flag\":false,
    \"forward_alerts\":false,
    \"last_updated\":1499425489,
    \"timezone\":\"Europe/London\",
    \"name\":\"Hostname; SVR_LON_3451\",
    \"updated_by\":5,
    \"description\":\"Downtime on host SVR_LON_3451\",
    \"id\":2,
    \"recurring_period_units\":0,
    \"start_date_time\":1499425462,
    \"timezone\" : \"Europe/London\"
  }
1
}

```

getMergeGroups

A Graze API GET request that returns details of all the custom merge groups in Cisco Crosswork Situation Manager.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getMergeGroups** does not take any request arguments.

Response

Endpoint **getMergeGroups** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
name	String	The merge group's name.
moolets	List	List of clustering algorithm Moolets to include in the custom merge group.
alert_threshold	Integer	Minimum number of alerts that must be present in a cluster before it can become a Situation. Must be greater than or equal to 1. Enter null if you want the custom merge group to use the default merge group value. Default merge group value is 2.
situation_similarity_limit	Floating Point	Percentage of alerts two Situations must share before they are merged for this group. Enter a value between 0 and 1. Entering null causes the merge group to use the same value as the default merge group.

Examples

The following examples demonstrate typical use of endpoint **getMergeGroups**:

Request example

Example cURL request to return all the custom merge groups in Cisco Crosswork Situation Manager:

```
curl -G
-u graze:graze
-k "https://example.com/graze/v1/getMergeGroups"
```

Response example

Example response returning details of all the custom merge groups in Cisco Crosswork Situation Manager:

```
[
{
  "name":"Default Cookbook",
  "moolets":
  [
    "Default Cookbook"
  ],
  "alert_threshold":2,
  "situation_similarity_limit":0.6
},
{
  "name":"Merge Group 1",
  "moolets":
  [
    "Recipe 1"
    "Recipe 2"
  ],
  "alert_threshold":null,
  "situation_similarity_limit":0.5
},
{
  "name":"Merge Group 2",
  "moolets":
  [
    "Recipe 2"
    "Time Based (Tempus)"
  ],
  "alert_threshold":2,
  "situation_similarity_limit":null
}
]
```

getPrcLabels

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getPrcLabels** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request.

			See the authenticate endpoint for more information.
sitn_id	Integer	Yes	Situation ID.
alert_ids	Number List	No	List of alert IDs.

Response

Endpoint **getPrcLabels** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getPrcLabels**:

Request example

Example cURL request to return the PRC labels for alerts 1, 2, 3, and 4 in Situation 157:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getPrcLabels?sitn_id=157&alert_ids=[1,2,3,4]"
```

Response example

Example response returning the PRC labels for alerts 1, 2, 3, and 4 in the Situation:

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

getProcesses

A Graze API GET request that returns a list of the processes in the database.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getProcesses** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
limit	Integer	No	Maximum number of processes to return. Defaults to 100.
exact_match	Boolean	No	If true , the query performs an exact match on the process name. If false , the query checks for contains only on the process name. Defaults to false .

Response

Endpoint **getProcesses** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
process_id	Number	ID of the process.
name	String	Name of the process.
description	String	Description of the process.

Examples

The following examples demonstrate typical use of endpoint **getProcesses**:

Request example

Example cURL request to return the first 100 processes containing "Network" in the process name:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getProcesses" --data-urlencode 'query=Network' --data-urlencode 'exact_match=false'
```

Response example

Example response returning three process names containing "Network":

```
[
  {
    "process_id": 1,
    "name": "Network LON",
    "description": "Network London"
  },
  {
    "process_id": 2,
    "name": "NY Network A",
    "description": "Network New York A"
  },
  {
    "process_id": 3,
```

```

    "name": "NY Network B",
    "description": "Network New York B"
  }
]

```

getRecipes

A Graze API GET request that returns all the Recipes in Cisco Crosswork Situation Manager.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getRecipes** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
cookbook	String	No	Name of the Cookbook that you want to return the Recipes for. Do not specify to return all Recipes.

Response

Endpoint **getRecipes** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Type	Description
List of JSON Objects	A list of all the Cookbooks in Cisco Crosswork Situation Manager and all their details.

Examples

The following examples demonstrate typical use of endpoint **getRecipes**:

Request example

Example cURL request to return all the Recipes in Cisco Crosswork Situation Manager:

```
curl -X GET -u graze:graze -k -v "https://localhost/graze/v1/getRecipes"
```

Response example

Example response returning all the Recipes in Cisco Crosswork Situation Manager:

```

[
  {
    "seed_alert_filter": "{}",
    "chef": "CValueRecipeV2",
    "description": "Alerts with a similar description",
    "recipe_alert_threshold": 2,
    "matcher": {
      "components": [
        {

```

```

        "similarity": 1,
        "name": "agent",
        "shingle_size": -1
    },
    {
        "similarity": 0.75,
        "name": "description",
        "shingle_size": -1
    }
]
},
"exclusion_filter": "{}",
"cook_for": null,
"max_sample_size": 10,
"rate": 0,
"min_sample_size": 5,
"cook_for_extension": 0,
"name": "Description",
"trigger_filter": "{ \"column\": \"severity\", \"op\": 5,
\"value\": 3, \"type\": \"LEAF\" }",
"cookbooks": [
    "Default Cookbook"
]
},
{
    "seed_alert_filter": "{}",
    "chef": "CValueRecipeV2",
    "description": "Alerts from a similar source",
    "recipe_alert_threshold": 2,
    "matcher": {
        "components": [
            {
                "similarity": 1,
                "name": "agent",
                "shingle_size": -1
            },
            {
                "similarity": 0.75,
                "name": "source",
                "shingle_size": 3
            }
        ]
    },
    "exclusion_filter": "{}",
    "cook_for": null,
    "max_sample_size": 10,
    "rate": 0,
    "min_sample_size": 5,
    "cook_for_extension": 0,
    "name": "Source",
    "trigger_filter": "{ \"column\": \"severity\", \"op\": 5,
\"value\": 3, \"type\": \"LEAF\" }",
    "cookbooks": [
        "Default Cookbook"
    ]
},

```

```

{
  "seed_alert_filter": "{}",
  "chef": "CValueRecipeV2",
  "description": "Remaining critical alerts",
  "recipe_alert_threshold": 1,
  "matcher": {
    "components": [
      {
        "similarity": 1,
        "name": "agent",
        "shingle_size": -1
      },
      {
        "similarity": 0.75,
        "name": "source",
        "shingle_size": 3
      }
    ]
  },
  "exclusion_filter": "{}",
  "cook_for": null,
  "max_sample_size": 10,
  "rate": 0,
  "min_sample_size": 5,
  "cook_for_extension": 0,
  "name": "Criticals",
  "trigger_filter": "{ \"column\": \"severity\", \"op\": 0,
  \"value\": 5, \"type\": \"LEAF\" }",
  "cookbooks": [
    "Default Cookbook"
  ]
}
]

```

getResolvingThreadEntries

A Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getResolvingThreadEntries** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
start	Number	No	Number of the first thread resolving entry to return. Default = 0.
limit	Number	No	Maximum number of resolving thread entries to return. Default = 100.

Response

Endpoint **getResolvingThreadEntries** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
sitn_id	Number	Situation ID.
resolving_entries	List	A list of resolving thread entries in the Situation. See below.

The **resolving_entries** list contains the following:

Name	Type	Description
entry_text	String	Text of the resolving entry. Reserved characters are converted to HTML entities, for example, & is converted to &amp; .
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.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getResolvingThreadEntries**:

Request example

Example cURL request to return 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 request to return 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"
```

Cisco Systems, Inc. www.cisco.com

Response example

Example response 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": "asdfsdf",
      "user_id": 3,
      "thread_name": "Support",
      "time": 1549385762,
      "entry_id": 1721
    },
    {
      "entry_text": "sdfsdf\n",
      "user_id": 3,
      "thread_name": "Support",
      "time": 1549385747,
      "entry_id": 1720
    }
  ]
}
```

getSecurityRealm

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

There are no other arguments because this endpoint returns data on all active security realms.

Response

Endpoint **getSecurityRealm** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
JSON Object	JSON	The name of the created Security Realms followed by its configured variables.

Examples

The following examples demonstrate typical use of endpoint **getSecurityRealm**:

Request example

Example cURL request 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"
```

Response example

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"}
}
```

getServices

A Graze API GET request that returns a list of the services in the database.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getServices** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
limit	Integer	No	Maximum number of services to return. Defaults to 1,000.
start	Integer	No	Number of the first service to return. Defaults to 0.
query	String	Yes	A JSON or SQL like filter of the service name.
exact_match	Boolean	No	If true , the query performs an exact match on the service name. If false , the query checks for contains only on the service name. Defaults to false .

Response

Endpoint **getServices** returns the following response:

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

Code	definitions for more information.
------	---

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

Name	Type	Description
service_id	Number	ID of the service.
name	String	Service name.
description	String	Description of the service.

Examples

The following examples demonstrate typical use of endpoint **getServices**:

Example Using Exact Matching

Example cURL request using exact matching of the query "Network LON":

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getServices" --data-urlencode 'query=Network LON' --data-urlencode 'exact_match=true'
```

Example response returning details of the service name "Network LON":

```
[{
  "service_id":3,
  "name":"Network LON",
  "description":"Network description"
}]
```

Example Using Approximate Matching

Example cURL request using approximate matching of the query "Network":

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getServices" --data-urlencode 'query=Network'
```

Example response returning details of all service names containing "Network":

```
[{
  "service_id":1,
  "name":"Network LON",
  "description":"Network London"
},{
  "service_id":2,
  "name":"NY Network A",
  "description":"Network New York A"
},{
  "service_id":3,
  "name":"NY Network B",
  "description":"Network New York B"
}]
```

getSeverities

A Graze API GET request that returns a list of possible severities and their severity IDs.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSeverities** takes the following request argument:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

There are no other arguments, as this endpoint returns data on all severities.

Response

Endpoint **getSeverities** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
name	String	Severity name.
severity_id	Number	ID of the severity.

Examples

The following examples demonstrate typical use of endpoint **getSeverities**:

Request example

Example cURL request to return the list of all severities:

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

Response example

Example response returning a list of all severities:

```
[{
  "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
}]
```

getSigCorrelationInfo

A Graze API GET request that returns all correlation information related to a specified Situation.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSigCorrelationInfo** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.

Response

Endpoint **getSigCorrelationInfo** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
sig_id	Number	Situation ID.
service_name	String	Service name.
external_id	String	External ID.
properties	List of Strings	Properties of the Situation.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSigCorrelationInfo**:

Request example

Example cURL request to return the correlation information for Situation ID 5:

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

Response example

Example response returning :

```
[
  {
    "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 Graze API GET request that returns a list of IDs of similar Situations, for a specified Situation ID and a limit.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSimilarSituationIds** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	ID of the Situation that you want to retrieve similar Situations for.
limit	Number	No	Maximum number of Situation IDs to return. Defaults to 100.

Response

Endpoint **getSimilarSituationIds** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
similarity_ids	Number List	List of IDs of similar Situations.
sig_id	Number	ID of the Situation that you requested to retrieve similar Situations for.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSimilarSituationIds**:

Request example

Example cURL request to return 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'
```

Response example

Example response returning the Situation IDs that are similar to Situation ID 1043:

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

getSimilarSituations

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSimilarSituations** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	ID of the Situation that you want to retrieve similar Situations for.
limit	Number	No	Maximum number of Situations to return. Defaults to 100.

Response

Endpoint **getSimilarSituations** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
similarities	Array	A list of Situations with similarity details. For each similar Situation: sim_sig_id : ID of the similar situation. similarity : Similarity value. 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.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSimilarSituations**:

Request example

Example cURL request to return Situations 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'
```

Response example

Example response returning details of two Situations, IDs 39 and 40, that are similar to Situation ID 38:

```
{
  "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 Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationActions** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_ids	Array	No, if from and to are used.	Array of Situation IDs that the actions are requested for.
start	Integer	Yes	Starting row from which data should be included in results.
limit	Integer	Yes	Maximum number of actions that you want to retrieve.
actions	Array	No	List of action codes. Returns all action codes if no action codes are specified. See Situation Action Codes for a list of action codes and their descriptions.
from	Number	No	Start time of the period you want to retrieve Situation actions for. This is a Unix epoch timestamp in seconds.
to	Number	No	End time of the period you want to retrieve Situation actions for. This is a Unix epoch timestamp in seconds.

Response

Endpoint **getSituationActions** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
activities	Array	An array of objects containing the Situation action details.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes

Closed Situation in historic database	Yes
---------------------------------------	-----

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint `getSituationActions`:

Request examples

Example cURL request 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'
```

Response example

Example response returning the actions for Situation 451:

```
{
  "activities": [
    {
      "uid": 2,
      "action_code": 1,
      "description": "Situation Created",
      "details": {},
      "type": "event",
      "sig_id": 451,
      "timed_at": 1507039842
    }, {
      "uid": 2,
      "action_code": 14,
      "description": "Added Alerts To Situation",
      "details": {}
      "alerts": [1, 2]
    }, {
      "uid": 3,
      "action_code": 11,
      "description": "Ran Tool",
      "details": {
        "tool_id": 271,
        "tool": "get data"
      },
      "sig_id": 451,
      "type": "event",
      "timed_at": 1557321088,
      "username": "admin"
    }
  ]
}
```

```
    1
  }
```

getSituationAlertIds

A Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationAlertIds** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
for_unique_alerts	Boolean	No	Indicates the alerts to return from the Situation: true = get only alerts unique to the Situation. false = get all alerts in the Situation. Default.

Response

Endpoint **getSituationAlertIds** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
total_alerts	Number	Total number of alerts, or unique alerts.
alert_ids	Number list	A list of the alert IDs.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationAlertIds**:

Request example

Example cURL request to return all the alert IDs for Situation ID 362:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationAlertIds" --data-urlencode
"sitn_id=362" --data-urlencode "for_unique_alerts=false"
```

Response example

Example response returning all the alerts for Situation ID 362:

```
{ "total_alerts": 232, "alert_ids": [ 6, 10, 17, 19, 22, 26, 27, 29, 32, 43, 44, 45, 47, 52, 6
7, 68, 79, 81, 83, 84, 96, 102, 105, 108, 109, 111, 113, 115, 116, 125, 135, 136, 138, 140, 142
, 143, 147, 151, 152, 153, 165, 175, 177, 178, 180, 181, 188, 192, 193, 207, 211, 213, 217, 22
3, 225, 232, 238, 239, 240, 244, 255, 258, 259, 269, 270, 272, 274, 284, 293, 303, 314, 318, 3
35, 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, 596, 599
, 601, 603, 628, 655, 656, 661, 664, 674, 684, 691, 705, 714, 715, 719, 720, 728, 732, 734, 75
0, 776, 777, 781, 788, 794, 808, 819, 830, 835, 838, 844, 857, 858, 860, 861, 877, 882, 885, 8
87, 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, 1238, 1254, 1271
, 1272, 1274, 1280, 1282, 1290, 1292, 1301, 1320, 1321, 1322, 1324, 1326, 1327, 1331, 1332
, 1333, 1362, 1379, 1402, 1414, 1423, 1433, 1443, 1454, 1468, 1472, 1473, 1481, 1491, 1510
, 1512, 1517, 1520, 1522, 1532, 1534 ] }
```

getSituationDescription

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationDescription** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.

Response

Endpoint **getSituationDescription** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
sitn_id	Number	Situation ID.
description	String	Text in the Situation' description field.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationDescription**:

Request example

Example cURL request to return the description for Situation ID 231:

```
curl -G-u graze:graze -k -v
"https://localhost/graze/v1/getSituationDescription" --data-urlencode
'sitn_id=231'
```

Response example

Example response returning the description for Situation ID 231:

```
{"sitn_id" : "231", "description" : "SyslogLamCookbook source"}
```

getSituationDetails

A Graze API GET request that returns the details of a specified Situation.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationDetails** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.

Response

Endpoint **getSituationDetails** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
category	String	Category of the alert.

created_at	Number	Time when Cisco Crosswork Situation Manager created the Situation. This is a Unix epoch timestamp in seconds.
custom_info	Object	Object containing the custom info for the Situation; null if there is no custom info for the Situation.
description	String	Description of the Situation.
first_event_time	Number	Time when Cisco Crosswork Situation Manager received the first event. This is a Unix epoch timestamp in seconds.
internal_priority	Number	Internal priority of the Situation.
last_event_time	Number	Time when Cisco Crosswork Situation Manager received the latest event. This is a Unix epoch timestamp in seconds.
last_state_change	Number	Time when the last state change occurred. This is a Unix epoch timestamp in seconds.
moderator_id	String	Owner of the Situation.
sitn_id	Number	Situation ID.
status	Number	Status of the Situation.
story_id	Number	
superseded_by	String	The ID of the Situation that supersedes this Situation, null if the Situation is not superseded.
total_alerts	Number	Total number of alerts in the Situation.
total_unique_alerts	Number	Total number of alerts that are unique to the Situation.
primary_team_id	Number	ID of the primary team assigned to the Situation. This is not returned if there is no primary team.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationDetails**:

Request example

Example cURL request to the details of Situation ID 173:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationDetails" --data-urlencode
'sitn_id=173'
```

Response example

Example response returning the details of Situation ID 173:

```
{
  "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": 173,
  "status": 1,
  "story_id": 3,
  "superseded_by": null,
  "total_alerts": 1403,
  "total_unique_alerts": 1403,
  "primary_team_id": 2
}
```

getSituationFlags

A Graze API GET request that returns the flags for one or an array of Situations.

See [Situation Flags](#) for more information on Cisco Crosswork Situation Manager Situation flags.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationFlags** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_ids	Array of Numbers	Yes	A list of Situation IDs.

Response

Endpoint **getSituationFlags** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Description
------	-------------

JSON List	List of the states those Situations
------------------	-------------------------------------

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationFlags**:

Request example

Example cURL request to list an array of all flags associated with Situation 1.

```
curl -X GET -u graze:graze -k -v
https://localhost/graze/v1/getSituationFlags?sitn_ids=%5B1%5D
```

Response example

Example response returning the flags associated with Situation 1:

```
{
  "1": [
    "NOTIFIED",
    "TICKETED"
  ]
}
```

getSituationHosts

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationHosts** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
for_unique_alerts	Boolean	No	Indicates the alerts to return from the Situation: true = get only alerts unique to the Situation. false = get all alerts in the Situation. Default.

Response

Endpoint **getSituationHosts** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationHosts**:

Request example

Example cURL request to return all the hosts that sent alerts to Situation ID 447:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getSituationHosts"
--data-urlencode 'sitn_id=447'
```

Response example

Example response returning all the hosts that sent alerts to Situation ID 447:

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


getSituationIds

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

Note: Take special care when using endpoint **getSituationIds**. Overuse of this endpoint can have a negative impact on the backend datastore.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationIds** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
query	String	Yes	A JSON or SQL like Situation filter.
limit	Number	No	Maximum number of Situation IDs to return.

Response

Endpoint **getSituationIds** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
total_situations	Number	Total number of Situations, or unique Situations.
sitn_ids	List	A list of Situation IDs.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationIds**:

Request example

Example cURL request to get the first 20 Situation IDs that match the query where the description is "lon_storage_636728" or the queue is 5:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getSituationIds" -
-data-urlencode 'query=description="lon_storage_636728" or queue = 5' --
data-urlencode 'limit=20'
```

Response example

Example response returning seven Situation IDs that match the query:

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

getSituationPrimaryTeam

A Graze API GET request that returns the primary team on the specified Situation.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationPrimaryTeam** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	ID of the Situation you want to retrieve the primary team for.

Response

Endpoint **getSituationPrimaryTeam** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
sitn_id	Number	ID of the Situation you wanted to retrieve the primary team for.
primary_team_id	Number	ID of the primary team for the Situation.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationPrimaryTeam**:

Request example

Example cURL request to return the primary team for Situation 1906:

```
curl -G -u graze:graze -k
"https://localhost/graze/v1/getSituationPrimaryTeam" --data-urlencode
'sitn_id=1906'
```

Response examples

Example response returning that team 36 is the primary team for Situation 1906:

```
{
  "primary_team_name": "Cloud DevOps",
  "sitn_id": 1906,
  "primary_team_id": 1
}
```

Example response returning that Situation 1906 does not have a primary team assigned to it:

```
{
  "sitn_id":1906,
}
```

getSituationProcesses

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationProcesses** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	ID of the Situation you want to return the process names for.

Response

Endpoint **getSituationProcesses** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
processes	List	A list of all the Situation's process names.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
------------------	-----------------------

Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationProcesses**:

Request example

Example cURL request to return all the process names for Situation 473:

```
curl -G -u graze:graze -k -v
"https://localhost/graze/v1/getSituationProcesses" --data-urlencode
'sitn_id=473'
```

Response example

Example response returning a list of all the Situation's process names:

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

getSituationServices

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationServices** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.

Response

Endpoint **getSituationServices** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

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

services	List	A list of the Situation's services.
-----------------	------	-------------------------------------

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationServices**:

Request example

Example cURL request to return the services for Situation ID 345:

```
curl -G -u graze:graze -k -v
'https://localhost/graze/v1/getSituationServices' --data-urlencode
'sitn_id=345'
```

Response example

Example response returning the services for the specified Situation:

```
{
  "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"
  ]
}
```

getSituationsWithFlag

A Graze API GET request that returns all the Situations which have the specified flag.

See [Situation Flags](#) for more information on Cisco Crosswork Situation Manager Situation flags.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationsWithFlag** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request.

			See the authenticate endpoint for more information.
flag	Number	Yes	Flag to search for.
start	Number	No	Starting point of the result set to return. Default is 0.
limit	Number	No	Number of results to return. Default is 1000.

Response

Endpoint **getSituationsWithFlag** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Type	Description
JSON Array	An array of the Situation IDs that have the specified flag associated with them.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationsWithFlag**:

Request example

Example cURL request to retrieve all Situations that have the specified flag associated with them.

```
curl -X GET -u graze:graze -k -v
https://localhost/graze/v1/getSituationsWithFlag?flag=NOTIFIED
```

Response example

Example response returning an array of all of the Situations that have the specified flag associated with them:

```
[
  1,
  2,
  5
]
```

getSituationTopology

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationTopology** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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 for more information on contextual hops. Vertex Entropy 0 = only nodes directly affected by the Situation. Default. 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: severity : Severity of the node. Severity Reference prc : Whether this node is the probable root cause of the alert. service : Service affected by the node. 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. description : Description of the node. 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

Endpoint **getSituationTopology** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return an array of JSON objects 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.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationTopology**:

Request 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, and description.

```
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"]
```

Response example

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSituationVisualization** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.

Response

Endpoint **getSituationVisualization** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return an array of JSON objects 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.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getSituationVisualization**:

Request example

Example cURL request to return 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"
```

Response Examples

Example response for a Situation created by a Cookbook Recipe:

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

Example response for a manually created Situation:

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

Example response when two Situations have been merged:

```
{
  "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
}
```

getStatuses

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getStatuses** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

There are no other arguments because this endpoint returns data on all statuses.

Response

Endpoint **getStatuses** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
status_id	Number	ID of the status.
name	String	Status name.

Examples

The following examples demonstrate typical use of endpoint **getStatuses**:

Request example

Example cURL request to return a list of statuses:

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

Response example

Example response returning a list of statuses:

```
[{
  "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 Graze API GET request that returns current system status information for all processes.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSystemStatus** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

There are no other arguments because this endpoint returns data on all processes.

Response

Endpoint **getSystemStatus** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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	Instance name.
last_heartbeat	Number	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	Number of missed process heartbeats. -1 is a special value indicating that a heartbeat has never been received.
process_name	String	Process name.
processes	List	A list of the processes, with status information.
reserved	Boolean	Indicates whether the process is reserved: true = a reserved process false = process that is not reserved
running	Boolean	Indicates whether the process is running: true = running false = not running
service_name	String	Service name.
display_name	String	Name of the service in the configuration.
type	String	Type of service, for example, lam, servlet, Moogfarmd.
passive	Boolean	Indicates whether the service is passive in a HA environment: true = passive false = active

stoppable	Boolean	Indicates whether the service is passive can be stopped: true = stoppable 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.

Examples

The following examples demonstrate typical use of endpoint **getSystemStatus**:

Request example

Example cURL request to return the system status:

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

Response example

Example response returning the system status:

```
{
  "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 Graze API GET request that returns a summary of current alerts and Situations in Cisco Crosswork Situation Manager.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getSystemSummary** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

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

Response

Endpoint **getSystemSummary** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object **system_summary**, containing the following statistics:

Name	Type	Description
total_events	Number	Total number of events in Cisco Crosswork Situation Manager.
open_sitns	Number	Number of open Situations in Cisco Crosswork Situation Manager.

open_sitns_up	Number	Number of open Situations that are trending up.
open_sitns_down	Number	Number of open Situations that are trending down.
avg_events_per_sitn	Number	Average number of events per Situation.
avg_alerts_per_sitn	Number	Average number of events per Situation.
service_count	Number	Number of services in Cisco Crosswork Situation Manager.
open_sigs_unassigned	Number	Number of unassigned Situations.

Examples

The following examples demonstrate typical use of endpoint **getSystemSummary**:

Request example

Example cURL request to return a summary of alerts and Situations in Cisco Crosswork Situation Manager:

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

Response example

Example response returning a summary of alerts and Situations in Cisco Crosswork Situation Manager:

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

getTeam

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getTeam** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
team_id	Integer	Yes	Unique ID of the team to retrieve information about.
name	String	Yes	Name of a valid team to retrieve information about.

Response

Endpoint **getTeam** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
JSON Object	JSON	Details of a specified team.

Examples

The following examples demonstrate typical use of endpoint **getTeam**:

Request examples

Example cURL request to return details of the team ID 1:

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

Example cURL request to return details of the team "Cloud DevOps":

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

Response example

Example response returning the details of the team:

```
{
  "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 Graze API GET request to return all teams related to the service with the specified ID or name.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getTeamsForService** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
service_id	String	No, if name is used.	ID of the service.
name	String	No, if service_id is used.	Name of the service.

Response

Endpoint **getTeamsForService** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
JSON Object	JSON	Details of the teams related to the specified service.

Examples

The following examples demonstrate typical use of endpoint **getTeamsForService**:

Request examples

Example cURL requests to return the teams related to service ID 1:

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

Example cURL requests to return the teams related to service "web":

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

Response example

Example response returning team and service details:

```
[
  {
    "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

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getTeamSituationIds** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
team_name	String	Yes	The name of an existing team.

Response

Endpoint **getTeamSituationIds** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
total_situation	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.

Examples

The following examples demonstrate typical use of endpoint **getTeamSituationIds**:

Request example

Example cURL request to return the Situations assigned to team "Cloud Devops":

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

Response example

Example response returning the total number of Situations followed by the ID of each situation.

```
{ "total_situations": 35, "sitn_ids": [ 20, 21, 39, 55, 85, 119, 145, 180, 208, 233, 244, 2
75, 305, 358, 460, 461, 485, 518, 574, 575, 592, 616, 666, 695, 696, 740, 800, 892, 919, 960,
992, 993, 1027, 1047, 1092 ] }
```

getTempus

A Graze API GET request that returns the details of all Tempus Moolets in Cisco Crosswork Situation Manager.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getTempus** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

There are no other arguments because this endpoint returns data on all Tempus Moolets in Cisco Crosswork Situation Manager.

Response

Endpoint **getTempus** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Type	Description
JSON Object	Names and configurations of all Tempus Moolets in Cisco Crosswork Situation Manager.

Examples

The following examples demonstrate typical use of endpoint **getTempus**:

Request example

Example cURL request to return the details of all Tempus Moolets:

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

Response example

Example response returning the details of Tempus algorithm "New Tempus 1":

```
[
  {
    "detection_algorithm": "Louvain",
    "minimum_arrival_similarity": 0.6667,
    "run_on_startup": true,
    "arrival_spread": 15,
    "execution_interval": 120,
    "description": "New Tempus 1",
    "alert_threshold": 4,
    "pre_partition": "",
    "partition_by": "",
    "window_size": 1200,
    "edge_weight": false,
    "significance_threshold": 1,
    "name": "Time Based (Tempus)",
    "entropy_threshold": 0.0,
    "process_output_of": [
      "Alert Workflows"
    ],
    "significance_test": "Poisson1",
    "bucket_size": 5
  }
]
```

getThreadEntries

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

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getThreadEntries** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
thread_name	String	Yes	Name of the thread to get entries from.
start	Number	No	Number of the first thread entry to return. Default is 0.
limit	Number	No	Maximum number of thread entries to return. Default is 100.

Response

Endpoint **getThreadEntries** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return an array of JSON objects 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 information:

Name	Type	Description
entry_text	String	Text of the thread entry. Reserved characters are converted to HTML entities, for example, & is converted to &amp; .
user_id	Number	User ID of the user that created the thread entry.
time	Number	Time when the thread entry was created. This is a Unix epoch timestamp in seconds.
entry_id	Number	ID of the thread entry.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	Yes

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getThreadEntries**:

Request example

Example cURL request to return 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"
```

Response example

Example response returning 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"
}
```

getToolShares

A Graze API GET request that returns the shared access for a specified tool.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getToolShares** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
tool_id	Number	Yes	ID of the tool that you want to retrieve its shared access for.

Response

Endpoint **getToolShares** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
tool_id	Number	ID of the tool you requested to return its shared access for.
domain_ids	Array	An array of all the IDs within the domain that can access the tool. If the domain is global, no domain IDs are returned.
domain	String	Domain that can access the tool. One of: user , team , role , or global .

Examples

The following examples demonstrate typical use of endpoint **getToolShares**:

Request example

Example cURL request to retrieve all the domain IDs that have access to tool 15:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/getToolShares" -H "Content-Type:
application/json; charset=UTF-8" -d '{"tool_id":15}'
```

Response example

Example response returning that tool ID 15 can be accessed by team ID 3:

```
{
  "tool_id": 15,
  "domain_ids": [
    3
  ],
  "domain": "team"
}
```

getTopPrcDetails

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

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, for the specified Situation, irrespective of whether they have been labeled causal or are unlabeled. Alerts marked as symptoms are excluded from the return.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getTopPrcDetails** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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 unlabeled alerts to return. Default is 1, if not specified, returning one alert with the highest root cause probability.

Response

Endpoint **getTopPrcDetails** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
rc_probability	Number	Root cause probability of the alert.
description	String	Description of the alert.

rc_label	Integer	Label defining whether the alert is causal or unlabeled. Alerts marked as symptoms are excluded from the return. 1 = causal 0 = unlabeled -1 = symptom
alert_id	Integer	Alert ID.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **getTopPrcDetails**:

Request example

Example cURL request to return 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

Example response returning the top three causal or unlabeled alerts for Situation ID 145:

```
{
  "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
  }
1
}

```

getUserInfo

A Graze API GET request that returns information about a specified user.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getUserInfo** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
user_id	Number	Yes	User ID of the the user to get information about.
username	String	Yes	A valid username.

Response

Endpoint **getUserInfo** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
user_id	Number	User ID.
full_name	String	Full name of the user.

Examples

The following examples demonstrate typical use of endpoint **getUserInfo**:

Request example

Example cURL request to return the information associated with user ID 57:

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

Response example

Example response returning the user information related to user ID 57:

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

getUserRoles

A Graze API GET request that returns the specified user's roles from the database.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getUserRoles** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
user_id	Number	No, if username is used.	User ID.
username	String	No, if user_id is used.	A valid username.

Response

Endpoint **getUserRoles** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
JSON Object	JSON	An array javascript object containing the role IDs, the role names and the role descriptions assigned to the user.

Examples

The following examples demonstrate typical use of endpoint **getUserRoles**:

Request example

Example cURL request to return the assigned roles for user "bigfish917":

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

Response example

Example response returning the roles assigned to the user:

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



```

    "description" : "Customer"
  }
]

```

getUsers

A Graze API GET request that returns a list of all users in the database.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getUsers** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
limit	Integer	No	The maximum number of results to return. Default is 1,000.

Response

Endpoint **getUsers** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
JSON Object	JSON	A JSON list of all users, displaying the user ID, teams, full name and username of each user.

Examples

The following examples demonstrate typical use of endpoint **getUsers**:

Request example

Example cURL request to return a maximum of three users:

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

Response example

Example response returning a maximum of three users:

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

```

```

    "uid": 6,
    "teams": [],
    "fullname": "Nagios",
    "username": "Nagios"
  },
  {
    "uid": 5,
    "teams": [],
    "fullname": "Webhook",
    "username": "Webhook"
  }
]

```

getUserTeams

A Graze API GET request that returns the team names and IDs associated with the specified user ID or username.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getUserTeams** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
user_id	Number	No, if username is used.	A valid user ID.
username	String	No if user_id is used.	A valid username.

Response

Endpoint **getUserTeams** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
JSON Object	JSON	A Javascript object containing the user ID and the teams that the user belongs to.

Examples

The following examples demonstrate typical use of endpoint **getUserTeams**:

Request example

Example cURL request to return the teams that user "admin" belongs to.

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

Response example

Example response returning the teams associated with username "admin":

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

getWorkflowEngineMoolets

A Graze API GET request that returns a list of Workflow Engine Moolets and the functions available in each. This endpoint returns an empty list if Moogfarmd is not running.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getWorkflowEngineMoolets** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

There are no other arguments because this endpoint returns data on all the Workflow Engine Moolets and the workflows associated with them.

Response

Endpoint **getWorkflowEngineMoolets** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Type	Description
List of JSON Objects	A list of Workflow Engine Moolets and information about them.

Examples

The following examples demonstrate typical use of endpoint **getWorkflowEngineMoolets**:

Request example

Example cURL request to return information on all the workflows in all the Workflow Engine Moolets in Cisco Crosswork Situation Manager:

```
curl -X GET -u graze:graze -k -v  
"https://localhost/graze/v1/getWorkflowEngineMoolets"
```

Response example

Example response returning information on all the Workflow Engine Moolets in Cisco Crosswork Situation Manager:

```
[  
  {  
    "active": true,  
    "last_updated": 1567420771,  
    "moolet_name": "Alert Workflows",  
    "functions": {  
      "alertInSituation": {  
        "decision": true,  
        "validators": null,  
        "name": "alertInSituation",  
        "description": "Check if the alert is in an active  
Situation.",  
        "arguments": [],  
        "actionOnAssociated": true,  
        "type": [  
          "alert"  
        ]  
      },  
      "alertNotInSituation": {  
        "decision": true,  
        "validators": null,  
        "name": "alertNotInSituation",  
        "description": "Check if the alert is not in an active  
Situation.",  
        "arguments": [],  
        "actionOnAssociated": true,  
        "type": [  
          "alert"  
        ]  
      },  
      "between": {  
        "decision": true,  
        "validators": null,  
        "name": "between",  
        "description": "Check to see if the trigger falls between  
two times, and optionally on specific days.",  
        "arguments": [  
          {  
            "name": "from",  
            "validator": {  
              "regex": "^[0-9]{2}:[0-9]{2}:[0-9]{2}$"
```

```

        },
        "description": "The 'from' time in hh:mm:ss 24hr
format",
        "type": "string",
        "required": true
    },
    {
        "name": "to",
        "validator": {
            "regex": "^[0-9]{2}:[0-9]{2}:[0-9]{2}$"
        },
        "description": "The 'to' time in hh:mm:ss 24hr
format",
        "type": "string",
        "required": true
    },
    {
        "name": "days",
        "description": "The optional list of days in short
form (Mon,Tue,Wed...), for all days use a blank list []",
        "type": "object",
        "required": true
    }
],
"actionOnAssociated": false,
"type": [
    "alert",
    "situation"
]
},
"contains": {
    "decision": true,
    "validators": null,
    "name": "contains",
    "description": "Check whether the specified object field
contains any of the listed values. Define values as an array, for example [
a ] or [ a, b, c ].",
    "arguments": [
        {
            "name": "field",
            "description": "The name of the object field to
check values in (including custom_info).",
            "type": "string",
            "required": true
        },
        {
            "name": "values",
            "description": "The list of values to check for,
any intersection is valid.",
            "type": "object",
            "required": true
        }
    ]
},
"actionOnAssociated": true,
"type": [
    "event",

```

```

        "alert",
        "situation"
    ]
},
"containsAll": {
    "decision": true,
    "validators": null,
    "name": "containsAll",
    "description": "Check whether the specified object field
contains all of the listed values. Define values as an array, for example [
a ] or [ a, b, c ].",
    "arguments": [
        {
            "name": "field",
            "description": "The name of the object field to
check values in (including custom_info).",
            "type": "string",
            "required": true
        },
        {
            "name": "values",
            "description": "The list of values to check for,
all must be included to be valid.",
            "type": "object",
            "required": true
        }
    ],
    "actionOnAssociated": true,
    "type": [
        "event",
        "alert",
        "situation"
    ]
},
"doesNotContain": {
    "decision": true,
    "validators": null,
    "name": "doesNotContain",
    "description": "Check whether the specified object field
does not contain any of the listed values. Define values as an array, for
example [ a ] or [ a, b, c ].",
    "arguments": [
        {
            "name": "field",
            "description": "The name of the object field to
check values in (including custom_info).",
            "type": "string",
            "required": true
        },
        {
            "name": "values",
            "description": "The list of values to check for,
any intersection will count.",
            "type": "object",
            "required": true
        }
    ]
}

```

```

        ],
        "actionOnAssociated": true,
        "type": [
            "event",
            "alert",
            "situation"
        ]
    },
    "moolet_type": "alert"
},
{
    "active": true,
    "last_updated": 1567420777,
    "moolet_name": "Enrichment Workflows",
    "functions": {
        "alertInSituation": {
            "decision": true,
            "validators": null,
            "name": "alertInSituation",
            "description": "Check if the alert is in an active
Situation.",
            "arguments": [],
            "actionOnAssociated": true,
            "type": [
                "alert"
            ]
        },
        "alertNotInSituation": {
            "decision": true,
            "validators": null,
            "name": "alertNotInSituation",
            "description": "Check if the alert is not in an active
Situation.",
            "arguments": [],
            "actionOnAssociated": true,
            "type": [
                "alert"
            ]
        },
        "between": {
            "decision": true,
            "validators": null,
            "name": "between",
            "description": "Check to see if the trigger falls between
two times, and optionally on specific days.",
            "arguments": [
                {
                    "name": "from"
,
                    "validator": {
                        "regex": "^[0-9]{2}:[0-9]{2}:[0-9]{2}$"
                    },
                    "description": "The 'from' time in hh:mm:ss 24hr
format",
                    "type": "string",
                    "required": true

```

```

    },
    {
      "name": "to",
      "validator": {
        "regex": "^[0-9]{2}:[0-9]{2}:[0-9]{2}$"
      },
      "description": "The 'to' time in hh:mm:ss 24hr
format",
      "type": "string",
      "required": true
    },
    {
      "name": "days",
      "description": "The optional list of days in short
form (Mon,Tue,Wed...), for all days use a blank list []",
      "type": "object",
      "required": true
    }
  ],
  "actionOnAssociated": false,
  "type": [
    "alert",
    "situation"
  ],
  "moolet_type": "alert"
},
]

```

getWorkflows

A Graze API GET request that returns workflows for a specified Moolet.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **getWorkflows** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
moolet_name	String	Yes	Name of the Moolet that you want to return the workflows for.

Response

Endpoint **getWorkflows** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
id	Integer	Unique ID of the workflow.

moolet_name	String	Name of the Moolet.																					
workflow_name	String	Name of the workflow.																					
description	String	Description of the workflow.																					
sequence	Integer	Sequence number of the workflow.																					
active	Boolean	Indicates whether or not the Moolet's associated Workflow Engine is active.																					
entry_filter	JSON	Filter to determine which events, alerts, or Situations can enter the workflow. If empty, the workflow accepts all events, alerts or Situations.																					
sweep_up_filter	JSON	Filter to intake any additional alerts or Situations from the database. Not relevant for event workflows.																					
first_match_only	Boolean	If enabled, alerts and Situations only pass through actions on the first time they enter the Workflow Engine. Not relevant for event workflows.																					
operations	JSON List	<p>List of properties relating to each operation:</p> <table> <thead> <tr> <th>Name</th><th>Type</th><th>Description</th></tr> </thead> <tbody> <tr> <td>type</td><td>String</td><td>Type of operation. Options are: 'action', 'decision' and 'delay'.</td></tr> <tr> <td>operation_name</td><td>String</td><td>Name of the operation. Only relevant for 'action' and 'decision' types.</td></tr> <tr> <td>function_name</td><td>String</td><td>Name of the function. Only relevant for 'action' and 'decision' types.</td></tr> <tr> <td>function_args</td><td>JSON Object</td><td>Arguments for the function.</td></tr> <tr> <td>duration</td><td>Integer</td><td>Length of time before the message goes to the next operation. Only relevant for 'delay' type.</td></tr> <tr> <td>reset</td><td>Boolean</td><td>Determines whether the timer resets after each occurrence. Only relevant for 'delay' type.</td></tr> </tbody> </table>	Name	Type	Description	type	String	Type of operation. Options are: 'action', 'decision' and 'delay'.	operation_name	String	Name of the operation. Only relevant for 'action' and 'decision' types.	function_name	String	Name of the function. Only relevant for 'action' and 'decision' types.	function_args	JSON Object	Arguments for the function.	duration	Integer	Length of time before the message goes to the next operation. Only relevant for 'delay' type.	reset	Boolean	Determines whether the timer resets after each occurrence. Only relevant for 'delay' type.
Name	Type	Description																					
type	String	Type of operation. Options are: 'action', 'decision' and 'delay'.																					
operation_name	String	Name of the operation. Only relevant for 'action' and 'decision' types.																					
function_name	String	Name of the function. Only relevant for 'action' and 'decision' types.																					
function_args	JSON Object	Arguments for the function.																					
duration	Integer	Length of time before the message goes to the next operation. Only relevant for 'delay' type.																					
reset	Boolean	Determines whether the timer resets after each occurrence. Only relevant for 'delay' type.																					

Examples

The following examples demonstrate typical use of endpoint **getWorkflows**:

Request example

Example cURL request to return workflows associated with the "Alert Workflows" Moolet:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/getWorkflows" -H
"Content-Type: application/json; charset=UTF-8" --data-urlencode
"moolet_name=Alert Workflows"
```

Response example

Example response returning information on the workflows associated with the "Alert Workflows" Moolet:

```
[
  {
    "id": 1,
    "moolet_name": "Alert 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

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **mergeSituations** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate

			endpoint for more information.
situations	List	Yes	An array of the Situation IDs you want to merge. Specify using Situation IDs, separating each item with a comma.
supersede_original	Boolean	Yes	Determines whether or not the original merged Situations are superseded by the new Situation.

Response

Endpoint **mergeSituations** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
sitn_id	Number	ID of the new merged Situation.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **mergeSituations**:

Request example

Example cURL request to merge Situations 31, 32, and 33 without superseding the original Situations by the new one:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/mergeSituations?auth_token=c4316d2cac524b96a1e4c787b68f7e3f&situations=%5B31%2C32%2C33%5D&supersede_original=false"
```

Response example

Example response returning the ID of the new merged Situation:

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

rateSituation

A Graze API POST request that applies a rating to a specified Situation.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **rateSituation** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sig_id	Long	Yes	ID of the Situation you want to rate.
rating	Integer	Yes	Rating that you want to apply to the Situation. This is equivalent to the number of stars that you can assign to a Situation in the UI. One of: 0 = Not yet rated 1 = Bad 2 = Poor 3 = Adequate 4 = Good 5 = Excellent
comment	String	No	A comment about the rating you are applying to the Situation.

Response

Endpoint **rateSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
rating	Number	Rating number applied to the Situation.
comment	String	Comment applied to the Situation.
sitn_id	Number	ID of the Situation that the rating was applied to.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **rateSituation**:

Request example

Example cURL request to apply a rating of 4 to Situation ID 18 with a comment "Rating 4":

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/rateSituation" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sig_id" : 18, "rating" : "4",
"comment" : "Rating 4"}'
```

Response example

Example response returning the rating, comment and Situation ID:

```
{"rating":4,"comment":"Rating 4","sitn_id":18}
```

removeAlertFromSituation

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **removeAlertFromSituation** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	ID of the alert you want to remove from the Situation.
sitn_id	Number	Yes	Situation ID.

Response

Endpoint **removeAlertFromSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

This endpoint does not remove the alert from the Situation if the alert has been archived to the historic database even if the Situation is still in the active database.

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **removeAlertFromSituation**:

Request example

Example cURL request to remove alert 16 from Situation 7:

```
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 }'
```

Cisco Systems, Inc. www.cisco.com

Response example

A successful request returns the HTTP code 200 and no response text.

removeSigCorrelationInfo

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **removeSigCorrelationInfo** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
service_name	String	No	Service name.
external_id	String	No	External ID.

Response

Endpoint **removeSigCorrelationInfo** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **removeSigCorrelationInfo**:

Request example

Example cURL request to remove the correlation information from Situation ID 3 for service name "my service 7" and external ID "my resource 7":

```
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"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

removeSituationPrimaryTeam

A Graze API POST request that removes the primary team from a Situation. The team remains assigned to the Situation.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **removeSituationPrimaryTeam** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	ID of the Situation that you want to remove the primary team from.

Response

Endpoint **removeSituationPrimaryTeam** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
sitn_id	Number	ID of the Situation that the primary team has been removed from.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **removeSituationPrimaryTeam**:

Request example

Example cURL request to remove the primary team from Situation 1906:

```
curl -G -u graze:graze -k
"https://localhost/graze/v1/removeSituationPrimaryTeam" --data-urlencode
'sitn_id=1906'
```

Response example

Example response returning the Situation ID that the primary team has been removed from:

```
{
  "sitn_id": 1906
}
```

reorderWorkflows

A Graze API POST request that reorders the sequence of workflows within a Workflow Engine Moolet.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **reorderWorkflows** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
moolet_name	String	Yes	Name of the Workflow Engine Moolet.
workflow_sequence	Array of Integers	Yes	An ordered array of all the workflow IDs within the Workflow Engine Moolet. The position of each workflow ID is its position within the Workflow Engine Moolet.

Response

Endpoint **reorderWorkflows** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **reorderWorkflows**:

Request example

Example cURL request to order the workflows within the "Alert Workflows" Workflow Engine as workflow ID 3 then workflow ID 1:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/reorderWorkflows" -H "Content-Type:
application/json; charset=UTF-8" --data '{
  "moolet_name" : "Alert Workflows",
  "workflow_sequences" : [3,1]
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

resolveAlerts

A Graze API POST request that resolves a list of alerts.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **resolveAlerts** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_ids	Number list	Yes	List of IDs of the alerts you want to resolve.
thread_entry_comment	String	No	Thread entry comment you want to add to the resolved alerts. HTML and XML tags are stripped from the thread entry text. Reserved characters are converted to HTML entities, for example, & is converted to &amp; .

Response

Endpoint **resolveAlerts** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **resolveAlerts**:

Request example

Example cURL request 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"}
```

Response example

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

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

resolveSituation

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **resolveSituation** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.

Response

Endpoint **resolveSituation** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **resolveSituation**:

Request example

Example cURL request to mark Situation ID 5 as resolved:

```
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}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

setAlertAcknowledgeState

A Graze API POST request that acknowledges or unacknowledges the owner of the specified alert ID.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setAlertAcknowledgeState** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	ID of the alert that you want to acknowledge or not acknowledge.
acknowledged	Number	Yes	The acknowledge state you want to apply to the alert: 0 for unacknowledged, 1 for acknowledged.

Response

Endpoint **setAlertAcknowledgeState** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	No
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setAlertAcknowledgeState**:

Request example

Example cURL request to set the acknowledge state of alert ID 7 to "acknowledged":

```
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 }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

setAlertSeverity

A Graze API POST request that sets the severity level of a specified alert.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setAlertSeverity** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
alert_id	Number	Yes	Alert ID.
severity	Number	Yes	The severity of the alert as an integer: 0 = Clear 1 = Indeterminate 2 = Warning 3 = Minor 4 = Major 5 = Critical

Response

Endpoint **setAlertSeverity** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	No
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setAlertSeverity**:

Request example

Example cURL request to set the alert with ID 7 as "Critical" :

```
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}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

setPrclabels

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

You can mark alerts as causal, non-causal or unlabeled within a Situation. An alert can have different PRC levels within different Situations.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setPrclabels** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
causal	Number list	Yes	A list of alert IDs that you want to be marked as causal.
non_causal	Number list	Yes	A list of alert IDs that you want to be marked as non-causal.
unlabelled	Number list	Yes	A list of alert IDs that you want to be marked as unlabeled.

Response

Endpoint **setPrclabels** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
------------------	-----------------------

Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setPrcLabels**:

Request example

Example cURL request to set alert ID 1 as causal, alert IDs 2 and 3 as non-causal, and alert 4 as unlabeled:

```
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]"
```

Response example

A successful request returns the HTTP code 200 and no response text.

setResolvingThreadEntry

A Graze API 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setResolvingThreadEntry** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
entry_id	Number	Yes	ID of the thread entry.
resolving_step	Boolean	Yes	Whether you are setting or clearing the thread entry as a resolving step.

Response

Endpoint **setResolvingThreadEntry** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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.
---------	--

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setResolvingThreadEntry**:

Request 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}'
```

Response example

Example response returning that the thread entry was successfully set as a resolving step:

```
true
```

setSituationAcknowledgeState

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setSituationAcknowledgeState** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
acknowledged	Number	Yes	The acknowledge state: 0 = unacknowledged 1 = acknowledged

Response

Endpoint **setSituationAcknowledgeState** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setSituationAcknowledgeState**:

Request example

Example cURL request to set the moderator on Situation ID 64 as acknowledged:

```
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 }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

setSituationDescription

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

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setSituationDescription** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
description	String	Yes	Description for the Situation ID.

Response

Endpoint **setSituationDescription** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setSituationDescription**:

Request example

Example cURL request to set the description for Situation ID 6 as " This is my description 12345" :

```
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"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

setSituationFlags

A Graze API POST request that updates the flags associated with a specified Situation. You can add flags to or remove them from a Situation.

See [Situation Flags](#) for more information on Cisco Crosswork Situation Manager Situation flags.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setSituationFlags** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_ids	Array of Numbers	Yes	An array of IDs for the Situations you want to update.
to_add	Array of Strings	Yes	Flags to be added to those Situations. If this is an empty list, no flags are added to the Situation.

to_remove	Array of Strings	Yes	Flags you want to remove from the Situation. If this is an empty list, no flags are removed from the Situation.
------------------	------------------	-----	---

Response

Endpoint **setSituationFlags** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setSituationFlags**:

Request example

Example cURL request to change the flags assigned to a situation. This change can include adding and/or removing flags. If one of the change arguments is left empty, nothing will change for that action.

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/setSituationFlags" -H "Content-Type:
application/json; charset=UTF-8" -d '{"sitn_ids":[ 1 ], "to_add": [
"NOTIFIED","TICKETED"],"to_remove": [ ] }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

setSituationPrimaryTeam

A Graze API POST request that sets one of the teams already assigned to a Situation as the primary team.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setSituationPrimaryTeam** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

sitn_id	Number	Yes	ID of the Situation.
team_id	Number	No, if team_name is used.	ID of the team that you want to make the primary team.
team_name	String	No, if team_id is used.	Name of the team that you want to make the primary team.

Response

Endpoint **setSituationPrimaryTeam** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
sitn_id	Number	ID of the Situation.
primary_team_id	Number	ID of the primary team.
primary_team_name	String	Name of the primary team.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setSituationPrimaryTeam**:

Request example

Example cURL request to set the team "Database Management System" as the primary team on Situation 1906:

```
curl -X POST -u graze:graze -k
"https://localhost/graze/v1/setSituationPrimaryTeam" -H "Content-Type:
application/json; charset=UTF-8" --data '{
  "sitn_id" : 1906,
  "team_name" : "Database Management System"
}'
```

Response example

Example response returning that team "Database Management System" is the primary team on Situation 1906:

```
{
  "sitn_id": 1906,
  "primary_team_id": 12,
  "primary_team_name": "Database Management System"
}
```

setSituationProcesses

A Graze API POST request that applies a list of processes to a specified Situation. Any other processes already associated with the Situation are removed.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setSituationProcesses** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
process_list	List	Yes	A list of process names as text strings. If any processes supplied do not exist in the database, the request creates them and assigns them to the Situation.

Response

Endpoint **setSituationProcesses** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setSituationProcesses**:

Request example

Example cURL request to set the processes for Situation ID as "Knowledge Management" and "90nm Manufacturing" :

```
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"]}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

setSituationServices

A Graze API POST request that applies a list of external services to a specified Situation. Any other services already associated with the Situation are removed.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **setSituationServices** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
sitn_id	Number	Yes	Situation ID.
service_list	List	Yes	A list of service names as text strings. If any services supplied do not exist in the database, they are created and assigned to the Situation.

Response

Endpoint **setSituationServices** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

API update behavior

The behavior of this endpoint depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This endpoint updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of endpoint **setSituationServices**:

Request example

Example cURL request to [complete]:

```
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"]}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

shareToolAccess

A Graze API POST request that shares access to a tool with other users, teams, or roles, or makes it global so that all users can access it. When a user creates a tool, it is automatically shared globally. You can use this endpoint to restrict its availability and ensure that tools are only available to users who need them. Using this endpoint to share access to a tool overwrites any existing shares.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **shareToolAccess** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
tool_id	Number	Yes	ID of the tool that you want to share access for.
domain	String	Yes	Domain to share access with. One of: user , team , role , or global .
domain_ids	Array	Yes/No	An array of one or more IDs within the domain. Optional for the global domain.

Response

Endpoint **shareToolAccess** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
domain_ids	Array	An array of the IDs for the domain that you requested to share tool access with.
domain	String	Domain that you requested to share access with. One of: user , team , role , or global .
tool_id	Number	ID of the tool you requested to share access with.

Examples

The following examples demonstrate typical use of endpoint **shareToolAccess**:

Request example

Example cURL request to share access of tool ID 15 with team ID 3:

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/shareToolAccess" -H "Content-Type:
application/json; charset=UTF-8" -d '{"tool_id":15, "domain":"team",
"domain_ids":[3]}'
```

Response example

Example response returning that the request to share access of tool ID 15 with team ID 3 was successful:

```
{
  "domain_ids": [
    3
  ],
  "domain": "team",
  "tool_id": 15
}
```

updateBotRecipe

A Graze API POST request that updates a Cookbook Bot Recipe.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateBotRecipe** takes the following request arguments. You must supply the name of the Bot Recipe plus at least one other argument that you want to change.

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the Recipe that you want to update.
cookbooks	Array of Strings	No	A list of the Cookbooks that this Recipe belongs to.
description	String	No	Description of the Recipe.
alert_threshold	Positive Integer	No	Minimum number of alerts required before Cookbook creates a Situation.
trigger	String	No	A filter that determines the alerts that Cookbook considers for Situation creation. Cookbook includes alerts that match the trigger filter. By default Cookbook only includes alerts with a

			severity of 'Critical'. For details on creating a filter, see /document/preview/35090#UIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy trigger filter, see /document/preview/114709#UID1bb978536bbf4215b156771ae826d901 for more information. Filter Search DataSet Up Vertex Entropy
exclusion	String	No	A filter that determines the alerts to exclude from Situation creation. Cookbook ignores alerts that match the exclusion filter. For details on creating a filter, see /document/preview/35090#UIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy exclusion filter, see /document/preview/114709#UID1bb978536bbf4215b156771ae826d901 for more information. Filter Search DataSet Up Vertex Entropy
seed_alert	String	No	A filter that determines whether to create a Situation from a seed alert. The seed alert must meet both the trigger , exclusion and seed_alert criteria to create a Situation. Cookbook considers subsequent alerts for clustering if they meet the trigger and exclusion filter criteria. Alerts that arrive prior to the seed alert that met the trigger and exclusion filter criteria do not form Situations. For details on creating a filter, see /document/preview/35090#UIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy seed alert filter, see /document/preview/114709#UID1bb978536bbf4215b156771ae826d901 for more information. Filter Search DataSet Up Vertex Entropy
rate	Positive Integer	No	Rate, in number of alerts per second. Cookbook clusters alerts if they arrive at a higher rate than is specified here. Cookbook uses rate together with min_sample_size and max_sample_size to determine whether to cluster alerts into Situations. See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples
min_sample_size	Positive Integer	No	Number of alerts that must arrive before the Cookbook starts to calculate the alert rate. See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Valid only if rate is non-zero. Cookbook and Recipe Examples
max_sample_size	Positive Integer	No	Maximum number of alerts that are considered in the alert rate calculation. When more than this number of alerts have arrived, Cookbook discards the oldest alerts and calculates the alert rate based on the number of alerts in the max_sample_size . See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Valid only if rate is non-zero. Cookbook and Recipe Examples
cook_for	Positive Integer	No	Minimum time period, in seconds, that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. See /document/preview/114512#UID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples If you set a different cook_for time for a Recipe, it overrides

			the Cookbook value. Recipes without a cook_for time inherit the value from the Cookbook.
cook_for_extension	Positive Integer	No	<p>Time period that Cookbook can extend clustering alerts for before the Recipe resets and starts a new cluster. Setting this value enables the cook for auto-extension feature for this Cookbook. As Cookbook receives related alerts, it continues to extend the total clustering time until the max_cook_for period is reached. Used in conjunction with the max_cook_for value, the cook_for_extension period helps to ensure that Cookbook continues to cluster alerts together that are related to the same failure. The cook_for_extension period only applies to new related alerts; it does not apply to existing alerts that are updated with new events. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information.Cookbook and Recipe Examples</p> <p>If you set a different cook_for_extension time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for_extension time inherit the value from the Cookbook.</p>
max_cook_for	Positive Integer	No	<p>Maximum time period that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. It works in conjunction with the cook_for_extension time to help ensure that Cookbook continues to cluster alerts together that are related to the same failure. This value is ignored unless the cook_for_extension time is specified. If cook_for_extension is set and this value is not set, the default is three times the cook_for value. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information.Cookbook and Recipe Examples</p> <p>If you set a different max_cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a max_cook_for value inherit the value from the Cookbook.</p>
cluster_by	String	No	<p>Determines Cookbook's clustering behavior. Set to an empty string to use the Cookbook cluster_by setting. Set to first_match so that Cookbook adds alerts to the first cluster over the similarity threshold value. Set to closest_match to add alerts to the cluster with the highest similarity greater than the similarity threshold value. This option may be less efficient because Cookbook needs to compare alerts against each cluster in a Recipe. Set to an empty string to use the Cookbook setting.</p> <p>If you set a different cluster_by value for a Recipe, it overrides the Cookbook value. Recipes without a cluster_by time inherit the value from the Cookbook.</p>
initialize_function	JSON Function Name	No	Default is initBuckets .
member_function	JSON Function	No	Default is checkBucket .

	on Name		
can_start_cluster	JSON Function Name	No	Default is null.
use_in_recipe	JSON Function Name	No	Default is null.
similarity	Double	No	Value between 0 and 1. Default is 0.8.

Response

Endpoint **updateBotRecipe** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateBotRecipe**:

Request example

Example cURL request to update the alert threshold to 4 in Bot Recipe "BotRecipe2":

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/updateBotRecipe" -H "Content-Type:
application/json; charset=UTF-8" -d
'{"name":"BotRecipe2","alert_threshold":4}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateCookbook

A Graze API POST request that updates a Cookbook.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateCookbook** takes the following request arguments. You must supply the name of the Cookbook plus at least one other argument that you want to change.

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

name	String	Yes	Name of the Cookbook that you want to update.
recipes	List of Strings	No	A list of the Recipes in this Cookbook. You must supply at least one Recipe. If you set first_recipe_match_only to first_match , Cookbook uses the order of the Recipes in this list to determine their priority. The first Recipe has the highest priority.
description	String	No	Description of the Cookbook.
process_output_of	List of Strings	No	Defines the source of the alerts that Cookbook processes. You can specify one or multiple Moolets. Valid values are: Alert Workflows , AlertBuilder , AlertRulesEngine , MaintenanceWindowManager , EmptyMoolet .
run_on_startup	Boolean	No	Whether Cookbook should start when Moogfarmd starts.
scale_by_severity	Boolean	No	Determines whether Cookbook ignores alerts with a severity of 0 (Clear). Set to true if you want Cookbook to ignore alerts with a severity of 0 (Clear). Set to false if you want Cookbook to include alerts with a severity of 0 (Clear).
entropy_threshold	Number	No	Minimum entropy value an alert must have in order for Cookbook to consider it for clustering it into a Situation. Cookbook does not include any alerts with an entropy value below the threshold in Situations.
first_recipe_match_only	Boolean	No	Defines whether Cookbook treats Recipes in priority order. If set to true , Cookbook adds an alert to a cluster created by the highest priority Recipe that meets the clustering criteria. The priority order is defined by the order of the Recipes in the recipes list. If set to false , Cookbook adds an alert to clusters in all the Recipes that meet the clustering criteria.
cluster_by	String	No	Determines Cookbook's clustering behavior. Set to first_match so that Cookbook adds alerts to the first cluster over the similarity threshold value. Set to closest_match to add alerts to the cluster with the highest similarity greater than the similarity threshold value. This option may be less efficient because Cookbook needs to compare alerts against each cluster in a Recipe. If you set a different cluster_by value for a Recipe, it overrides the Cookbook value. Recipes without a cluster_by time inherit the value from the Cookbook.
cook_for	Integer	No	Minimum time period, in seconds, that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples

			If you set a different cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for time inherit the value from the Cookbook.
cook_for_extension	Integer	No	<p>Time period that Cookbook can extend clustering alerts for before the Recipe resets and starts a new cluster. Setting this value enables the cook for auto-extension feature for this Cookbook. As Cookbook receives related alerts, it continues to extend the total clustering time until the max_cook_for period is reached. Used in conjunction with the max_cook_for value, the cook_for_extension period helps to ensure that Cookbook continues to cluster alerts together that are related to the same failure. The cook_for_extension period only applies to new related alerts; it does not apply to existing alerts that are updated with new events. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples</p> <p>If you set a different cook_for_extension time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for_extension time inherit the value from the Cookbook.</p>
max_cook_for	Integer	No	<p>Maximum time period that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. It works in conjunction with the cook_for_extension time to help ensure that Cookbook continues to cluster alerts together that are related to the same failure. This value is ignored unless the cook_for_extension time is specified. If cook_for_extension is set and this value is not set, the default is three times the cook_for value. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples</p> <p>If you set a different max_cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a max_cook_for value inherit the value from the Cookbook.</p>
moobot	String	No	<p>The Moobot you want Cookbook to use if there are any Bot Recipes. See /document/preview/114188#UUID1ecd7d867ea666f30327761f670221b7 for more information. Recipe Types</p>

Response

Endpoint **updateCookbook** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateCookbook**:

Request example

Example cURL request to update Cookbook "GrazeCookBook1":

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/updateCookbook" -H "Content-Type:
application/json; charset=UTF-8" -d '{"name" : "GrazeCookBook1",
"run_on_startup":true, "cook_for_extension":7200, "max_cook_for":14400}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateDefaultMergeGroup

A Graze API POST request that updates the default merge group in Cisco Crosswork Situation Manager.

Clustering algorithms, such as Cookbook and Tempus, use the default values in the default merge group unless you have set up custom merge groups with different values to merge Situations from these clustering algorithms. You can set up merge groups using the UI (see </document/preview/121633#UUIIDdc5f5ef14beb1148529d6f5b50806b63> for details) or using the Graze API endpoint [addMergeGroup](#). Configure Merge Groups

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateDefaultMergeGroup** takes the following request arguments:

Name	Type	Required	Description
alert_threshold	Integer	No	Minimum number of alerts that must be present in a cluster before it can become a Situation in the merge group. Must be greater than or equal to 1. Default value is 2.
situation_similarity_limit	Floating Point	No	Percentage of alerts two Situations must share before they are merged. A value between 0 and 1. Default value is 0.7.

Response

Endpoint **updateDefaultMergeGroup** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateDefaultMergeGroup**:

Request example

Example cURL request to set the default merge group's **alert_threshold** to 1:

```
curl -X POST
-u graze:graze -
k -v "https://example.com/graze/v1/updateDefaultMergeGroup"
-H "Content-Type: application/json; charset=UTF-8"
-d '{
  "alert_threshold":1
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateMaintenanceWindow

A Graze API POST request that updates an existing maintenance window.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateMaintenanceWindow** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
window_id	String	Yes	ID of the maintenance window that you want to update.
name	String	No	Name of the maintenance window.
description	String	No	Description of the maintenance window.
filter	String	No	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	Unix epoch time in seconds (Number)	No	Start time of the maintenance window. This must be in Unix epoch time and may be up to 5 years in the future.
duration	Seconds (Number)	No	Duration of the maintenance window in seconds. The minimum duration is 1 second and the maximum is 157784630 seconds (5 years).
forward_alerts	Boolean	No	Whether or not alerts should be forwarded to the next Moollet in the processing chain.
recurring_period	Number	No	Whether or not this is a recurring maintenance window. Set this to 1 for a recurring maintenance window. 0 for a one-time maintenance window. If not specified, default is 0 . If you set this property 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. Valid values are: 2 = daily 3 = weekly 4 = monthly Default is 0 if recurring_period is set to 0 .
timezone	String	No	Time zone that you want the maintenance window to be in. You can only change the time zone if the maintenance window is inactive when you make the request. The time zone must be a valid entry in the IANA Time Zone Database . When scheduling recurring maintenance windows, Cisco Crosswork Situation Manager takes into account any daylight savings time changes for the time zone.

Response

Endpoint **updateMaintenanceWindow** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateMaintenanceWindow**:

Request examples

Example cURL request to update all the parameters in the existing maintenance window ID 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 ID 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}'
```

Example cURL request to update the existing maintenance window ID 144 to be in time zone "Europe/London":

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/updateMaintenanceWindow" -H "Content-Type:
application/json; charset=UTF-8" -d '{"window_id":144, "timezone" :
"Europe/London"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateMergeGroup

A Graze API POST request that updates a custom merge group in Cisco Crosswork Situation Manager.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateMergeGroup** takes the following request arguments:

Name	Type	Required	Description
name	String	Yes, along with at least one other argument.	The custom merge group's name.
moolets	List	No	List of clustering algorithm Moolets to include in the custom merge group.
alert_threshold	Integer	No	Minimum number of alerts that must be present in a cluster before it can become a Situation. Must be greater than or equal to 1. Enter null if you want the custom merge group to use the default merge group value. Default merge group value is 2.
situation_similarity_limit	Floating Point	No	Percentage of alerts that two Situations in this merge group must share before they are merged. A value between 0 and 1. Enter null if you want the merge group to use the default merge group value. Default merge group value is 0.7.

Response

Endpoint **updateMergeGroup** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateMergeGroup**:

Request example

Example cURL request to update a custom merge group's **situation_similarity_limit**:

```
curl -X POST
-u graze:graze -
k -v "https://example.com/graze/v1/updateMergeGroup"
-H "Content-Type: application/json; charset=UTF-8"
-d '{
  "name": "Merge Group 1",
```



```
"situation_similarity_limit":0.6
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateSecurityRealm

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

Warning: Warn any users who are 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.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateSecurityRealm** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the security realm.
type	String	Yes	Security realm type. This must be "SAML2" .
active	Boolean	Yes	Determines whether the new realm is active or not.
configuration	JSON Object	Yes	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 /document/preview/11687#UUID34c1daabda9eb021bd2a4ba43eda683b.Security Configuration Reference

Response

Endpoint **updateSecurityRealm** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateSecurityRealm**:

Request example

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=
```

Cisco Systems, Inc. www.cisco.com

```

{
  "idpMetadata": "<?xml version='1.0' encoding='UTF-
8'><EntitiesDescriptor Name='urn:keycloak'
xmlns='urn:oasis:names:tc:SAML:2.0:metadata'><EntityDescriptor
entityID='http://moogsaml:18080/auth/realms/master'><IDPSSODescri
ptor
WantAuthnRequestsSigned='true'><nprotocolSupportEnumeration='urn:oasis:
names:tc:SAML:2.0:protocol'><KeyDescriptor
use='signing'><dsig:KeyInfo><dsig:KeyName>l8ddhI8SroeNnlq0TkTxIj2
VI-
0bvr2QfG_o32jWeKI</dsig:KeyName><dsig:X509Data><dsig:X509Certifica
te>MIICmzCCAYMCBgFk8A9vMjANBgkqhkiG9w0BAQsFADARMQ8wDQYDVQQDDAZtYXN0ZXIwHhcN
MTgwNzMxMTEwNjQwWhcnMjgwNzMxMTEwODIwWjARMQ8wDQYDVQQDDAZtYXN0ZXIwggEiMA0GCSq
GSIB3DQEBAQUAA4IBDwAwggEKAoIBAQC0Iiz3dBu696slyduAb1BMuvR1bmdTKVBMICWaeEcS8R
zw8gWthPQpw2e202LjOeu4VktVMEEAUa2IrLS4QpYgyhOuzapcIGF4kB0AREbalWa7C9od9%2Be
TqWgvXPrDokzp7g%2B%2Ba5yvtKxE3ieUORPPACvLWcbkMwyb%2Be5V8%2Bz8n4263Uol8srSax
Lsm/oTozJNwbG%2BbzV8JQHU3xFV5nFbyNySvc%2B/B7tDFZuJC5BMu6bwi/rPqp5OMcuB1W
%2BxCcX7IYPphnBjRWNYQJD3gRckjrurjISkTEcqpZEjr79isbofQaPDi5TSjglPD5rr0OWMVqv9
1a1/pVN2y0y%2BR1T8HAgMBAAEWdQYJKoZIhvcNAQELBQADggEBAAgRhWYKESVsTRAUVYzHYpt
d3/ex47%2BTVXhjPO0ORLUJbhtfhgohtyejd6ohazkcSgMy6%2BwaeVojqq4Q/tzCOW2EAQO9
QOQdaBWOPxDXhJ9TGQJE2E28SS2Gg6paAMfRmtA7c6xXii%2BYfLo3PG1SSc/sGe4KIPKflkqq
DEqEeaY1olPZU2bLnpMSIui2nK1crE2%2Bt9apLWAGosah6scMGZ9vTrtOvrNuhB2LuU3cvRQWr
UBaQuXQsBV7Q6a8lkrzZ6rjAibO4vcEL4yjqPnA%2BhetuhBlGPQj6ntuhdrnmoKmWYY97wk8eXw
blhQxg8GUyfqabfOAKwiGAKlxgkexm20M=</dsig:X509Certificate></dsig:X509D
ata></dsig:KeyInfo></dsig:KeyDescriptor></dsig:SingleLogoutService>
<Binding='urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST'><Location='http://moogsaml:18080/auth/realms/master/protoco
l/saml'>
</dsig:SingleLogoutService><Binding='urn:oasis:names:tc:SAML:2.0:bindin
gs:HTTP-
Redirect'><Location='http://moogsaml:18080/auth/realms/master/prot
ocol/saml'>
</dsig:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent</NameIDFormat><NameIDFormat>urn:oasis:names:tc:SAML:
2.0:nameid-
format:transient</NameIDFormat><NameIDFormat>urn:oasis:names:tc:SAML:1
.1:nameid-
format:unspecified</NameIDFormat><NameIDFormat>urn:oasis:names:tc:SAML
:1.1:nameid-format:emailAddress</NameIDFormat><SingleSignOnService
Binding='urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST'><Location='http://moogsaml:18080/auth/realms/master/protoco
l/saml'>
</dsig:SingleSignOnService><Binding='urn:oasis:names:tc:SAML:2.0:bindin
gs:SOAP'><Location='http://moogsaml:18080/auth/realms/master/prot
ocol/saml'>
</dsig:IDPSSODescriptor></dsig:EntityDescriptor></dsig:EntitiesDescripto
r>",
  "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"
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateTeam

Request that updates an existing team.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateTeam** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
team_id	Number	Yes	Team ID.
name	String	No	Team name. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is.
alert_filter	String	No	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	No	List of the team services names or IDs. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is.
sig_filter	String	No	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	No	Team default landing page. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is.
active	Boolean	No	False if the team is inactive, true if the team is active. Default to true. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is.
description	String	No	Team description. Exclude this attribute to leave Cisco Crosswork Situation Manager as it is.
users	List of numbers or strings	No	Team users (either IDs or usernames). Exclude this attribute to leave Cisco Crosswork Situation Manager as it is.

Response

Endpoint **updateTeam** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateTeam**:

Request example

Example cURL request to update the information for team ID 16:

```
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" : []}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateTempus

A Graze API POST request that updates an existing Tempus Moolet.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateTempus** takes the following request arguments. You must supply the name of the Tempus algorithm plus at least one other argument that you want to change.

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the Tempus algorithm. Must be unique.
description	String	No	Description of the Situations Tempus generates.
process_output_of	List	Yes	Defines the source of the alerts that Tempus processes.
run_on_startup	Boolean	No	Whether this Tempus algorithm should start when Moogfarmd starts.
entropy_threshold	Number	No	Minimum entropy value for an alert to be clustered into a Situation. Tempus does not cluster any alerts with an entropy value below the threshold into Situations.
execution_interval	Number	No	Executes Tempus after a defined number of seconds.
window_size	Number	No	Determines the length of time when Tempus analyzes alerts and clusters them into a Situation each time it runs.

bucket_size	Number	No	Determines the time span of each bucket in which alerts are captured. Default bucket size is 5 seconds.
minimum_arrival_similarity	Number	No	How similar alerts must be to be considered for clustering.
alert_threshold	Number	No	<p>Minimum number of alerts that match the clustering criteria before the Tempus algorithm creates a Situation.</p> <p>When Tempus determines the number of alerts required to create a Situation, it compares the alert threshold values in Tempus and in the merge group that Tempus belongs to, and it uses the higher value. If you are using the default merge group which has an alert threshold of 2, Tempus will never create a Situation containing a single alert. If you want Cisco Crosswork Situation Manager to create Situations with a single alert, consider changing the alert threshold in the default merge group to 1 or creating custom merge groups. See /document/preview/121633#UIDdc5f5ef14beb1148529d6f5b50806b63 for more information on updating the default merge group and setting up custom merge groups. Configure Merge Groups</p>
partition_by	String	No	<p>Splits clustering according to the entered component. After alerts have been clustered and before they enter merging and resolution, you can split clusters into sub-clusters based on a component of the events. For example, you can use the manager parameter to ensure that Situations only contain events from the same manager.</p> <p>Note: Cisco does not recommend partitioning by components.</p>
pre_partition	Boolean	No	Partitions event streams before clustering. You specify a component field on which the event stream will be partitioned before clustering occurs. The alerts in the resulting Situations each contain a single value for the component field chosen.
significance_test	String	No	<p>Calculation that determines how significant a cluster of alerts or a potential Situation must be for Tempus to detect it. Poisson1, looks at the data of a single alert cluster to calculate how significant it is. This more likely to detect all significant alert clusters but with a higher risk of creating insignificant alert clusters. Use this option when your alerts originate from different networks or unrelated topologies. Poisson2 is a more thorough test that looks at an alert cluster and all alerts outside the cluster with a similar event rate. It is more likely to exclude all insignificant alert clusters but with a high risk of excluding significant alert clusters. Use this option if you expect all of your alerts to come from the same connected network. See Poisson distribution for more information.</p>

significance_threshold	Number	No	Sets the maximum significance score for Tempus to create a Situation. The score is proportional to the probability that the alert cluster or potential Situation was coincidence. The lower the score, the more significant the cluster and the least likely it was a coincidence. This score ranges from 0 to 100.
detection_algorithm	String	No	Detection algorithm that Tempus uses, one of: Louvain , LouvainMulti , or SmartLocal .

Response

Endpoint **updateTempus** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateTempus**:

Request example

Example cURL request to update the detection algorithm on Tempus algorithm "newTempus":

```
curl -X POST -u graze:graze -k "https://localhost/graze/v1/updateTempus" -H
"Content-Type: application/json; charset=UTF-8" --data '{ "name" :
"newTempus", "detection_algorithm": "LouvainMulti" }'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateUser

A Graze API POST request that updates an existing user.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateUser** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
username	String	No, if uid is used.	Username of the user to be updated.
uid	Long	No, if username is used.	User ID of the user to be updated.
password	String	No	New user password, only valid for DB realm.
active	Boolean	No	true if the user is active, false if the user is

			inactive. Defaults to true .
email	String	No	User's email address.
fullname	String	No	User's full name.
roles	JSON list	No	List of either the role IDs or the role names. For example, "roles":["Super User"] .
primary_group	String or Number	No	User's primary group name or primary group ID.
department	String or Number	No	User's department ID or department name.
timezone	String	No	User's timezone.
contact_num	String	No	User's phone number.
session_expiry	Number	No	Number of minutes after which the user's session expires. Defaults to the system default.
competencies	JSON list	No	A list with the user competencies. Each competency should have name or cid and ranking. For example: <pre>[{ "name": "SunOS", "ranking": 40 }, { "name": "SAP", "ranking": 50 }, { "name": "EMC", "ranking": 60 }]</pre>
teams	JSON list of Numbers or Strings	No	List of the user's team names or team IDs.

Response

Endpoint **updateUser** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateUser**:

Request example

Example cURL request to update user ID 5:

```
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"], "session_expiry" : null, "properties" : null,
```

```
"contact_num" : "555-123456", "timezone" : "Europe/London", "fullname" :
"John Doe", "department" : "Support", "primary_group" : "Network", "email"
: "test@test.com"}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateValueRecipe

A Graze API POST request that updates a Cookbook Recipe that uses either a Value Recipe or a Value Recipe v2 recipe type. See </document/preview/114188#UUID1ecd7d867ea666f30327761f670221b7> for more information. Recipe Types

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateValueRecipe** takes the following request arguments. You must supply the name of the Cookbook Recipe plus at least one other argument that you want to change.

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
name	String	Yes	Name of the Recipe that you want to update.
cookbook	List of Strings	No	A list of the Cookbooks that this Recipe belongs to.
description	String	No	Description of the Recipe.
version	String	No	Defines whether the Recipe uses Value Recipe or Value Recipe v2. Valid values are V1 for the Value Recipe and V2 for Value Recipe v2. Default is V2. See /document/preview/114188#UUID1ecd7d867ea666f30327761f670221b7 for more information. Use addBotRecipe if you want to create a Bot Recipe. Recipe Types
alert_threshold	Positive Integer	No	Minimum number of alerts required before Cookbook creates a Situation. When Cookbook determines the number of alerts required to create a Situation, it compares the alert threshold values in the Cookbook Recipe and in the merge group that the Cookbook Recipe belongs to, and it uses the higher value. If you are using the default merge group which has an alert threshold of 2, Cookbook will never create a Situation containing a single alert. If you want Cisco Crosswork Situation Manager to create Situations with a single alert, consider changing the alert threshold in the default merge group to 1 or creating custom merge groups. See /document/preview/121633#UUIDdc5f5ef14beb1148529d6f5b50806b63 for more information on updating the default merge group and setting up custom merge groups. Configure Merge Groups

trigger	String	No	A filter that determines the alerts that Cookbook considers for Situation creation. Cookbook includes alerts that match the trigger filter. By default Cookbook only includes alerts with a severity of 'Critical'. For details on creating a filter, see /document/preview/35090#UUIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy trigger filter, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d901 for more information. Filter Search DataSet Up Vertex Entropy
exclusion	String	No	A filter that determines the alerts to exclude from Situation creation. Cookbook ignores alerts that match the exclusion filter. For details on creating a filter, see /document/preview/35090#UUIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy exclusion filter, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d901 for more information. Filter Search DataSet Up Vertex Entropy
seed_alert	String	No	A filter that determines whether to create a Situation from a seed alert. The seed alert must meet both the trigger , exclusion and seed_alert criteria to create a Situation. Cookbook considers subsequent alerts for clustering if they meet the trigger and exclusion filter criteria. Alerts that arrive prior to the seed alert that met the trigger and exclusion filter criteria do not form Situations. For details on creating a filter, see /document/preview/35090#UUIDf7925c4a2878b75931b6f34600f25045 . To set a vertex entropy seed alert filter, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d901 for more information. Filter Search DataSet Up Vertex Entropy
rate	Positive Integer	No	Rate, in number of alerts per second. Cookbook clusters alerts if they arrive at a higher rate than is specified here. Cookbook uses rate together with min_sample_size and max_sample_size to determine whether to cluster alerts into Situations. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples
min_sample_size	Positive Integer	No	Number of alerts that must arrive before the Cookbook starts to calculate the alert rate. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Valid only if rate is non-zero. Cookbook and Recipe Examples
max_sample_size	Positive Integer	No	Maximum number of alerts that are considered in the alert rate calculation. When more than this number of alerts have arrived, Cookbook discards the oldest alerts and calculates the alert rate based on the number of alerts in the max_sample_size . See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Valid only if rate is non-zero. Cookbook and Recipe Examples
cook_for	Positive Integer	No	Minimum time period, in seconds, that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples If you set a different cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for time inherit the value from

			the Cookbook.
cook_for_extension	Positive Integer	No	<p>Time period that Cookbook can extend clustering alerts for before the Recipe resets and starts a new cluster. Setting this value enables the cook for auto-extension feature for this Cookbook. As Cookbook receives related alerts, it continues to extend the total clustering time until the max_cook_for period is reached. Used in conjunction with the max_cook_for value, the cook_for_extension period helps to ensure that Cookbook continues to cluster alerts together that are related to the same failure. The cook_for_extension period only applies to new related alerts; it does not apply to existing alerts that are updated with new events. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples</p> <p>If you set a different cook_for_extension time for a Recipe, it overrides the Cookbook value. Recipes without a cook_for_extension time inherit the value from the Cookbook.</p>
max_cook_for	Positive Integer	No	<p>Maximum time period that Cookbook clusters alerts for before the Recipe resets and starts a new cluster. It works in conjunction with the cook_for_extension time to help ensure that Cookbook continues to cluster alerts together that are related to the same failure. This value is ignored unless the cook_for_extension time is specified. If cook_for_extension is set and this value is not set, the default is three times the cook_for value. See /document/preview/114512#UUID8eba89afa7e23d6e518a695de750d106 for more information. Cookbook and Recipe Examples</p> <p>If you set a different max_cook_for time for a Recipe, it overrides the Cookbook value. Recipes without a max_cook_for value inherit the value from the Cookbook.</p>
cluster_by	String	No	<p>Determines Cookbook's clustering behavior. Set to an empty string to use the Cookbook cluster_by setting. Set to first_match so that Cookbook adds alerts to the first cluster over the similarity threshold value. Set to closest_match to add alerts to the cluster with the highest similarity greater than the similarity threshold value. This option may be less efficient because Cookbook needs to compare alerts against each cluster in a Recipe. Set to an empty string to use the Cookbook setting.</p> <p>If you set a different cluster_by value for a Recipe, it overrides the Cookbook value. Recipes without a cluster_by time inherit the value from the Cookbook.</p>
matcher	JSON Structure	No	<p>A JSON structure containing:</p> <p>hop_limit: Maximum number of hops between the alert source nodes in order for the alerts to qualify for clustering. Cisco Crosswork Situation Manager measures hop limit from the first alert that formed the Situation and always follows the shortest possible route in the network. A hop is the jump between two directly connected nodes in a network. For more information on hops, see /document/preview/113737#UUID6cc2007bfdb8ce5fe9adef8dbcb8307d. To set a hop limit, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d90</p>

			<p>1 for more information. Vertex Entropy Set Up Vertex Entropy</p> <p>You can only use a hop limit if you have imported your network topology into the system. See /document/preview/11702#UUID517934cfa987aa330c16e11e1be74be8 for details. Import a Topology</p> <p>components: Values that alerts must match for Cookbook to include them in a Situation. You can provide multiple values such as source, description, service or custom info fields. An array of JSON objects, each containing:</p> <table> <tr> <th>Name</th><th>Type</th><th>Required</th><th>Description</th></tr> <tr> <td>name</td><td>String</td><td>Yes</td><td>Name of the component.</td></tr> <tr> <td>similarity</td><td>Double</td><td>Yes</td><td>Similarity threshold that the component must meet for Cookbook to cluster the alert into a Situation.</td></tr> <tr> <td>shingle_size</td><td>Integer</td><td>No</td><td>Shingle size for Cookbook to use to determine the similarity between different strings. The shingle size is only valid for Recipe Value v2 recipes. Default is -1 which means that Cookbook uses words to determine similarity. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for more details. Recipe Types</td></tr> <tr> <td>treat_as</td><td>String</td><td>No</td><td>Determines whether Cookbook treats the component as a string or matches each value in the list individually. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for details. Valid values are List and String. Default is String. Recipe Types</td></tr> <tr> <td>case_sensitive</td><td>Boolean</td><td>No</td><td>Enables or disables case sensitive when comparing strings. Case sensitivity is only valid for Recipe Value recipes. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for more details. Default is true which means that strings are treated as case sensitive. Recipe Types</td></tr> </table>	Name	Type	Required	Description	name	String	Yes	Name of the component.	similarity	Double	Yes	Similarity threshold that the component must meet for Cookbook to cluster the alert into a Situation.	shingle_size	Integer	No	Shingle size for Cookbook to use to determine the similarity between different strings. The shingle size is only valid for Recipe Value v2 recipes. Default is -1 which means that Cookbook uses words to determine similarity. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for more details. Recipe Types	treat_as	String	No	Determines whether Cookbook treats the component as a string or matches each value in the list individually. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for details. Valid values are List and String . Default is String . Recipe Types	case_sensitive	Boolean	No	Enables or disables case sensitive when comparing strings. Case sensitivity is only valid for Recipe Value recipes. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for more details. Default is true which means that strings are treated as case sensitive. Recipe Types
Name	Type	Required	Description																								
name	String	Yes	Name of the component.																								
similarity	Double	Yes	Similarity threshold that the component must meet for Cookbook to cluster the alert into a Situation.																								
shingle_size	Integer	No	Shingle size for Cookbook to use to determine the similarity between different strings. The shingle size is only valid for Recipe Value v2 recipes. Default is -1 which means that Cookbook uses words to determine similarity. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for more details. Recipe Types																								
treat_as	String	No	Determines whether Cookbook treats the component as a string or matches each value in the list individually. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for details. Valid values are List and String . Default is String . Recipe Types																								
case_sensitive	Boolean	No	Enables or disables case sensitive when comparing strings. Case sensitivity is only valid for Recipe Value recipes. See /document/preview/11781#UUID9f0ccf5294ab3a0a056e74b025174d43 for more details. Default is true which means that strings are treated as case sensitive. Recipe Types																								

Response

Endpoint **updateValueRecipe** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateValueRecipe**:

Request example

Example cURL request to update Value Recipe "Value onTwo":

```
curl -X POST -u graze:graze -k -v
"https://localhost/graze/v1/updateValueRecipe" -H "Content-Type:
application/json; charset=UTF-8" --data '{
  "name" : "Source",
  "hop_limit" : 2,
  "components" : [{ "name": "thenewvalue", "similarity":0.8,
"shingle_size" : 3}]
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

updateWorkflow

A Graze API POST request that updates an existing workflow in the Workflow Engine.

Back to [Graze API EndPoint Reference](#).

Request arguments

Endpoint **updateWorkflow** takes the following request arguments:

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
id	String	Yes	ID of the workflow that you want to update.
workflow_name	String	No	Name of the workflow.
active	Boolean	No	Determines whether the workflow is active or not.
description	String	No	Description of the workflow.
entry_filter	JSON	No	Filter to determine which events, alerts or Situations can enter the workflow. You can use an SQL-like query. See /document/preview/124174#UIDbd640e729795a2b9ead884be59a6e23f for more information on creating SQL-like filters. If empty, the workflow accepts all events, alerts or Situations. Filter Search Data
sweep_up_filter	JSON	No	Filter to intake any additional events, alerts or Situations from the database. You can use an SQL-like query. See /document/preview/124174#UIDbd640e729795a2b9ead884be59a6e23f for more information on creating SQL-like filters. Filter Search Data
first_match_only	Boolean	No	If enabled, events, alerts, and Situations only pass through

nly	n		actions on the first time they enter the Workflow Engine.			
operations	JSON List	No	List of properties relating to each operation:			
			Name	Type	Required	Description
			type	String	Yes	Type of operation. Options are: 'action', 'decision' and 'delay'.
			operation_name	String	Yes, for 'action' and 'decision' types.	Name of the operation.
			function_name	String	Yes, for 'action' and 'decision' types.	Name of the function.
			forwarding_behavior	String	No	Forwarding behavior for the function. One of: always forward : The function always forwards the object to the next workflow. stop this workflow : The function stops this workflow and the object moves to the next workflow. stop all workflows : The function stops all workflows for this object.Default is always forward . Only valid for 'action' and 'decision' types.
			function_args	JSON	No	Arguments for

				Object		the function.
			duration	Integer	Yes, for 'delay' type.	Length of time before the message goes to the next operation.
			reset	Boolean	Yes, for 'delay' type.	Determines whether the timer resets after each occurrence.

Response

Endpoint **updateWorkflow** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Examples

The following examples demonstrate typical use of endpoint **updateWorkflow**:

Request examples

Example cURL request to deactivate workflow id 14:

```
curl -X POST -u graze:graze -k \
-v "https://localhost/graze/v1/updateWorkflow" \
-H "Content-Type: application/json; charset=UTF-8" \
--data '{
  "id" : 14,
  "active" :false
}'
```

Example cURL request to rename workflow id 14:

```
curl -X POST -u graze:graze -k \
-v "https://localhost/graze/v1/updateWorkflow" \
-H "Content-Type: application/json; charset=UTF-8" \
--data '{
  "id" : 14,
  "workflow_name" : "Deactivated Example"
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

Integrations API

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

Contact Cisco Support if you experience difficulties or need further guidance.

Endpoints

See [Integrations API Endpoint Reference](#) for details of all the Integrations API endpoints.

API definition

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

https://<server>/graze/v1/<endpoint>

Examples:

https://localhost/graze/v1/integrations/

https://localhost/graze/v1/integrations/{integrationId}

https://localhost/graze/v1/integrations/{integrationId}/status

Authentication

In order to use the Integrations API, you must have the `manage_integrations` permission. See [/document/preview/35141#UUID1b6353d943aee6691d57d631ed647220](#) for more information. Role Permissions

All requests require a basic authentication header.

Integrations API Endpoint Reference

This is a reference list for the Integrations API endpoints. Follow the links to see the details of each endpoint.

All of the endpoints use basic authorization.

Brokers

The following endpoints relate to brokers:

- [/broker-profiles](#) Create a Broker Profile

Integrations

The following endpoints relate to integration management:

- [/integrations](#)
- [/integrations/{integrationId}](#)
- [/integrations/{integrationId}/status](#)

/integrations

The **/integrations** endpoint allows you to create new integrations.

To read and update existing integrations see [/integrations/{integrationId}](#).

Back to [Integrations API Endpoint Reference](#).

POST

Creates an integration's configuration.

Request arguments

The POST request takes the following request arguments:

Name	Type	Required	Description
type_id	String	Yes	Type of integration to add, for example Webhook
inputs	String	Yes	The key and value of inputs to substitute into the integration's configuration. This can include username, password, URL to poll, and timing intervals.
name	String	Yes	Name of the integration to add, for example Webhook1
version	String	Yes	Version of the integration to use. For validation purposes, this must be the most recent version.

Response

The POST request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
readonly	List	Read-only details about the integration. This can include the webhook URL of the integration and authorisation details.
type_id	String	Type of integration you have created.
inputs	String	Username (value) and password (key) you have configured to authenticate with the integration.
name	String	Name of the integration you have created.
id	Integer	ID of the integration you have created.
version	String	Version of the integration you have created.
config	Object	The integration's configuration.

Examples

The following examples demonstrate typical use making a POST request to the endpoint **integrations**:

Request example

Example cURL POST request to create a Webhook integration:

```
curl -X POST \
https://example.com/integrations/api/v1/integrations \
-u John.Doe:MyPassword \
-d '{
  "type_id": "Webhook",
  "inputs": [
    {
```



```
    "name": "username",
    "value": "<username>"
  },
  {
    "name": "password",
    "value": "<integrationpassword>"
  }
],
"name": "Webhook1",
"version": "1.14"
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

Example response returning the new Webhook integration's configuration:

```
{
  "id": 4,
  "name": "Webhook2",
  "type_id": "Webhook",
  "version": "1.14",
  "config": {
    "category": "monitoring",
    "description": "A webhook integration to allow events to be sent
via generic REST and processed by Moogsoft AIOps.",
    "display_name": "Webhook",
    "type_id": "Webhook",
    "version": "1.14",
    "config": {
      "monitor": {
        "name": "Webhook Lam Monitor",
        "class": "CRestMonitor",
        "port": "$config#port()",
        "authentication_type": "basic_auth_static",
        "basic_auth_static": {
          "username": "John.Doe",
          "password": "Password123"
        }
      },
      "use_ssl": false,
      "accept_all_json": true,
      "lists_contain_multiple_events": true,
      "num_threads": 5,
      "rest_response_mode": "on_receipt",
      "rpc_response_timeout": 20
    },
    "constants": {
      "severity": {
        "CLEAR": 0,
        "INDETERMINATE": 1,
        "WARNING": 2,
        "MINOR": 3,
        "MAJOR": 4,
        "CRITICAL": 5,
        "0": 0,
        "1": 1,

```

```
        "2": 2,
        "3": 3,
        "4": 4,
        "5": 5,
        "moog_lookup_default": 1
    }
},
"conversions": {
    "sevConverter": {
        "input": "STRING",
        "output": "INTEGER",
        "lookup": "severity"
    },
    "stringToInt": {
        "input": "STRING",
        "output": "INTEGER"
    }
},
"filter": {
    "presend": "WebhookLam-SolutionPak.js",
    "modules": [],
    "dependencies": {
        "lambot": [
            "LamBot.js",
            "WebhookLam-SolutionPak.js"
        ],
        "contrib": []
    }
},
"mapping": {
    "lambotOverridden": [],
    "catchAll": "overflow",
    "rules": [
        {
            "name": "signature",
            "rule": "$source::$type"
        },
        {
            "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"
        }
    ]
}
},
"ha_profile": "active_active"
},
"inputs": [
    {
        "key": "username",
        "value": "password"
    }
],
"readonly": [
    {
        "name": "url",
        "description": "URL:",
        "value":
"https://example.com/integrations/api/v1/events/webhook2"
    },
    {
        "name": "userid",
        "description": "User ID:",
        "value": "<username>"
    },
    {
        "name": "readonly_password",
        "description": "Password:",
        "value": "<password>"
    },
    {

```

```

    "name": "auth",
    "description": "Base64 Encoded Auth:",
    "value": "Basic YWRtaW46"
  }
]
}

```

[/integrations/{integrationId}](#)

The **`/integrations/{integrationId}`** endpoint allows you to read and update existing integrations.

To create a new integration see [/integrations](#).

Back to [Integrations API Endpoint Reference](#).

GET

Retrieves a specific integration's configuration.

Request arguments

The GET request takes the following request argument:

Name	Type	Required	Description
integrationId	Integer	Yes	ID of the integration to retrieve. You can obtain an integration's ID by executing a GET request to <code>/integrations</code> .

There are no other arguments because this endpoint returns data based on the integration ID alone.

Response

The GET request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
readonly	List	Read-only details about the integration. This can include the webhook URL of the integration and authorization details.
type_id	String	Type of integration.
inputs	String	Username (value) and password (key) you have configured to authenticate with the integration.
name	String	Name of the integration.
id	Integer	ID of the integration.
version	String	Version of the integration. For validation purposes, this must be the most recent version.
config	Object	Integration's configuration.

Examples

The following examples demonstrate making a GET request to the endpoint **integrations/{integrationId}**:

Request example

Example cURL request for details of the integration with the ID "3":

```
curl \
https://example.com/integrations/api/v1/integrations/3 \
-u John.Doe:MyPassword \
```

Response example

A successful request returns the HTTP code 200 and no response text.

Example response returning the integration's details:

```
{
  "id": 3,
  "name": "DynatraceAPMPolling1",
  "type_id": "dynatrace_apm_lam",
  "version": "2.3",
  "config": {
    "config": {
      "filter": {
        "modules": [],
        "presend": "DynatraceApmLam.js",
        "dependencies": {
          "lambot": [
            "LamBot.js",
            "DynatraceApmLam.js"
          ],
          "contrib": []
        }
      },
      "mapping": {
        "rules": [
          {
            "name": "signature",
            "rule": "$systemprofile :: $rule"
          },
          {
            "name": "source_id",
            "rule": "Dynatrace APM"
          },
          {
            "name": "external_id",
            "rule": "$id"
          },
          {
            "name": "manager",
            "rule": "Dynatrace Apm"
          },
          {
            "name": "source",
            "rule": "$source"
          }
        ]
      }
    }
  }
}
```

```

        "name": "class",
        "rule": "$rule"
    },
    {
        "name": "agent",
        "rule": "$LamInstanceName"
    },
    {
        "name": "agent_location",
        "rule": "$LamInstanceName"
    },
    {
        "name": "type",
        "rule": "$state"
    },
    {
        "name": "severity",
        "rule": "$severity",
        "conversion": "sevConverter"
    },
    {
        "name": "description",
        "rule": "$message"
    },
    {
        "name": "agent_time",
        "rule": "$start",
        "conversion": "timeConverter"
    }
],
"catchAll": "overflow",
"lambotOverridden": [
    "custom_info.overflow",
    "source"
]
},
"monitor": {
    "name": "DynatraceApm Lam Monitor",
    "class": "CDynatraceApmMonitor",
    "targets": {
        "target1": {
            "url": "https://localhost:8021",
            "filter": {
                "state": "InProgress",
                "profileName": "nam",
                "incidentRule": "rul"
            },
            "timeout": 120,
            "password": "def",
            "username": "abc",
            "disable_certificate_validation": true
        }
    },
    "max_retries": -1,
    "retry_interval": 60,
    "request_interval": 60
}

```

```

    },
    "constants": {
        "severity": {
            "severe": 5,
            "warning": 2,
            "informational": 1
        }
    },
    "conversions": {
        "stringToInt": {
            "input": "STRING",
            "output": "INTEGER"
        },
        "sevConverter": {
            "input": "STRING",
            "lookup": "severity",
            "output": "INTEGER"
        },
        "timeConverter": {
            "input": "STRING",
            "output": "INTEGER",
            "timeFormat": "yyyy-MM-dd'T'HH:mm:ss.SSS"
        }
    }
},
"moolets": [],
"type_id": "dynatrace_apm_lam",
"version": "2.3",
"category": "monitoring",
"ha_profile": "active_passive",
"description": "An integration which enables Moogsoft AIOps to
ingest events from Dynatrace APM.",
"display_name": "Dynatrace APM (Polling)"
},
"inputs": [
    {
        "key": "targets",
        "value": [
            {
                "url": "https://localhost:8021",
                "filter": {
                    "state": "InProgress",
                    "profile_name": "nam",
                    "incident_rule": "rul"
                },
                "password": "def",
                "username": "abc"
            }
        ]
    }
],
{
    "key": "timing",
    "value": {
        "timeout": 120,
        "retry_interval": 60,
        "request_interval": 60
    }
}

```

```

    }
  },
  "readonly": null
}

```

PUT

Updates an integration's configuration. Integrations with the ID you specify are unavailable during the update. When the update completes, they automatically resume.

Request arguments

The PUT request takes the following request argument:

Name	Type	Required	Description
integrationId	Integer	Yes	ID of the integration to update.
type_id	String	Yes	Type of integration to add, for example Webhook .
inputs	String	Yes	The key and value of inputs to substitute into the integration's configuration. This can include username, password, URL to poll, and timing intervals.
name	String	Yes	Name of the integration to add, for example Webhook1 .
version	String	Yes	Version of the integration to use. For validation purposes, this must be the most recent version.

Response

The PUT request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
readonly	List	Read-only details about the integration. This can include the webhook URL of the integration and authorisation details.
type_id	String	Type of integration you have updated.
inputs	String	Username (value) and password (key) you have configured to authenticate with the integration.
name	String	Name of the integration you have updated.
id	Integer	ID of the integration you have updated.
version	String	Version of the integration you have updated.
config	Object	The integration's configuration.

Examples

The following examples demonstrate making a PUT request to the endpoint **integrations/{integrationId}**:

Request example

Example cURL PUT request to update the **value** and **password** parameters for a Webhook integration. In this example, the Webhook's ID is **2**:

```
curl -X PUT \
https://example.com/integrations/api/v1/integrations/2 \
-u John.Doe:MyPassword \
-d '{
  "type_id": "Webhook",
  "inputs": [
    {
      "name": "username",
      "value": "Jane.Doe"
    },
    {
      "name": "password",
      "value": "Password123"
    }
  ],
  "name": "Webhook1",
  "version": "1.14"
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

Example response returning the updated integration's details:

```
{
  "id": 2,
  "name": "Webhook2",
  "type_id": "Webhook",
  "version": "1.14",
  "config": {
    "category": "monitoring",
    "description": "A webhook integration to allow events to be sent
via generic REST and processed by Moogsoft AIOps.",
    "display_name": "Webhook",
    "type_id": "Webhook",
    "version": "1.14",
    "config": {
      "monitor": {
        "name": "Webhook Lam Monitor",
        "class": "CRestMonitor",
        "port": "$config#port()",
        "authentication_type": "basic_auth_static",
        "basic_auth_static": {
          "username": "<username>",
          "password": "<password>"
        }
      },
      "use_ssl": false,
```

```
    "accept_all_json": true,
    "lists_contain_multiple_events": true,
    "num_threads": 5,
    "rest_response_mode": "on_receipt",
    "rpc_response_timeout": 20
  },
  "constants": {
    "severity": {
      "CLEAR": 0,
      "INDETERMINATE": 1,
      "WARNING": 2,
      "MINOR": 3,
      "MAJOR": 4,
      "CRITICAL": 5,
      "0": 0,
      "1": 1,
      "2": 2,
      "3": 3,
      "4": 4,
      "5": 5,
      "moog_lookup_default": 1
    }
  },
  "conversions": {
    "sevConverter": {
      "input": "STRING",
      "output": "INTEGER",
      "lookup": "severity"
    },
    "stringToInt": {
      "input": "STRING",
      "output": "INTEGER"
    }
  },
  "filter": {
    "presend": "WebhookLam-SolutionPak.js",
    "modules": [],
    "dependencies": {
      "lambot": [
        "LamBot.js",
        "WebhookLam-SolutionPak.js"
      ],
      "contrib": []
    }
  },
  "mapping": {
    "lambotOverridden": [],
    "catchAll": "overflow",
    "rules": [
      {
        "name": "signature",
        "rule": "$source::$type"
      },
      {
        "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"
        }
    ]
}
},
"ha_profile": "active_active"
},
"inputs": [
    {
        "key": "Jane.Doe",
        "value": "MyPassword"
    }
],
"readonly": [
    {

```

```

        "name": "url",
        "description": "URL:",
        "value":
"https://example.com/integrations/api/v1/events/webhook2"
    },
    {
        "name": "userid",
        "description": "User ID:",
        "value": "Username"
    },
    {
        "name": "readonly_password",
        "description": "Password:",
        "value": "Password123"
    },
    {
        "name": "auth",
        "description": "Base64 Encoded Auth:",
        "value": "Basic YWRtaW46"
    }
  ]
}

```

/integrations/{integrationId}/status

The **/integrations/{integrationID}/status** endpoint allows you to check and update the status of an integration.

Back to [Integrations API Endpoint Reference](#).

GET

Retrieves the status of an integration.

Request arguments

The GET request takes the following request argument:

Name	Type	Required	Description
integrationId	Integer	Yes	ID of the integration to retrieve. You can obtain an integration's ID by executing a GET request to /integrations .

Response

The GET request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
instances	String	URL of the instance(s) in an Object of Broker IDs that map to the integration status info. For example: <pre> "instances": { </pre>

		<pre>"Broker_3a52b7ef_8bad_41cc_8b36_cbb9c2aa3a9e@: { "integration_name": "Azure1", "status": "running", "last_heartbeat": 1567789762645 }</pre>
global	Object	Contains integration_config_id and status .
integration_config_id	Integer	ID of the integration.
status	String	Current status of the integration.

Examples

The following examples demonstrate making a GET request to the endpoint `/integrations/{integrationId}/status`:

Request example

Example cURL GET request to retrieve the current status of the integration with the ID **"6"**:

```
curl \
https://example.com/integrations/api/v1/integrations/6/status \
-u John.Doe:MyPassword \
-H "accept: application/json"
```

Response example

A successful request returns the HTTP code 200 and no response text.

Example response returning the integration's current status:

```
{
  "global":
  {
    "integration_config_id": 6,"
    "status": "running"
  },
  "instances":
  {
    "Broker_3a52b7ef_8bad_41cc_8b36_cbb9c2aa3a9e":
    {
      "integration_name": "Azure1",
      "status": "running",
      "last_heartbeat": 1567789762645
    }
  }
}
```

PUT

Updates the status of an integration. You can use this to start and stop integrations. An integration can run on multiple brokers; when assigning an integration to a broker, Cisco Crosswork Situation Manager favours the broker(s) running the least integrations.

Request arguments

The PUT request takes the following request argument:

Cisco Systems, Inc. www.cisco.com

Name	Type	Required	Description
integrationId	Integer	Yes	ID of the integration to update.

Response

The PUT request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Required	Description
command	String	Yes	Status command you want to make. Choose from start and stop .
target	String	Yes, unless using the start command	Broker on which to run or stop the integration. If unspecified, Cisco Crosswork Situation Manager starts the integration on the broker running the least integrations.

Examples

The following examples demonstrate making a PUT request to the endpoint **/integrations/{integrationId}/status**:

Request example

Example cURL PUT request to update an integration.

```
curl -X PUT \
https://example.com/integrations/api/v1/integrations/6/status \
-u John.Doe:MyPassword \
{
  "command": "start"
}
```

Response example

A successful request returns the HTTP code 204 and no response text.

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

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

Situation Flags

You can use Situation flags to determine actions that users have performed on Situations, such as adding a manual description to a Situation or manually assigning a Situation to a team.

The table below shows the list of codes and the matching description for each Situation flag available in Cisco Crosswork Situation Manager:

Flag Code	Description
1	LEAVE_MANUAL_DESCRIPTION
2	MANUALLY_ASSIGNED_TO_TEAM

You can use the following Graze API endpoints and MoogDb V2 methods to create more Situation flags and associate them with Situations. For example, you may want to set up a flag for "TICKETED" when a ticket has been raised for a Situation.

- **checkSituationFlag**: Checks whether a flag is associated with a Situation, in [Graze API](#) and [MoogDb V2](#).
- **getSituationFlags**: Returns the flags for one or an array of Situations, in [Graze API](#) and [MoogDb V2](#).
- **getSituationsWithFlag**: Returns all the Situations which have the specified flag, in [Graze API](#) and [MoogDb V2](#).

- **setSituationFlags**: Updates the flags associated with a specified Situation, in [Graze API](#) and [MoogDb V2](#).

API Update Behavior

The behavior of the Graze API endpoints and MoogDb V2 methods depends on the status of the Situation they are acting on. The three relevant statuses are:

- Open Situation: The Situation is open.
- Closed Situation in active database: For a period of time after the Situation has been closed, it remains in the active database. This period of time is known as the "grace period".
- Closed Situation in historic database: After the grace period has expired, the Situation is moved to the historic database.

Each API endpoint/method topic describes its behavior in these three Situation statuses.

See </document/preview/11704#UUID828885544e984d640fbcf75089ac6732> for more information on the active and historic databases. [Configure Historic Data Retention](#)

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](#): Returns the number of alerts that belong to new Situations in the specified time range.
- [getMTTASStats](#): Returns the Mean Time To Acknowledge (MTTA) Situations in the specified time range.
- [getMTTDStats](#): Returns the Mean Time To Detect (MTTD) Situations in the specified time range.
- [getMTTRStats](#): Returns the Mean Time To Resolve (MTTR) for Situations in the specified time range.
- [getNewAlertsStats](#): Returns the number of new alerts in the specified time range.
- [getNewAlertsPerSituationsStats](#): Returns the percentage of noise reduction from alerts-to-Situations clustering in the specified time range.
- [getNewEventsPerAlertsStats](#): Returns the percentage of noise reduction from events-to-alerts aggregation and deduplication in the specified time range.
- [getNewEventsPerSituationsStats](#): Returns the percentage of noise reduction from events-to-Situations aggregation, deduplication, and clustering in the specified time range.
- [getNewSituationsStats](#): Returns the number of new Situations created in the specified time range.

- [getReassignedSituationStats](#): Returns the number of Situations reassigned in the specified time range.
- [getReoccurringSituationStats](#): Returns the percentage of reoccurring situations in the specified time range.
- [getServiceSituationStats](#): Returns the number of active Situations impacting a service in the specified time range.
- [getSeveritySituationStats](#): Returns the number of Situations by severity in the specified time range.
- [getStats](#): Returns all available Stats API endpoints along with their description and request parameters.
- [getStatusSituationStats](#): Returns the number of Situations by status.
- [getSystemSituationStats](#): Returns the number of active Situations in the specified time range.
- [getTopServiceSituationStats](#): Returns the number of active Situations impacting a top service in the specified time range.

Team Endpoints

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

- [getCommentCountPerTeamStats](#): Returns the total number of comments each hour for a specific team or teams in the specified time range.
- [getMTTAPerTeamStats](#): Returns the mean time to acknowledge (MTTA) a Situation per team in the specified time range.
- [getMTTRPerTeamStats](#): Returns the mean time to resolve (MTTR) a Situation per team in the specified time range.
- [getReassignedSituationsPerTeamStats](#): Returns the number of reassigned Situations associated with a team or multiple teams in the specified time range.
- [getReoccurringSituationPerTeamStats](#): Returns the number of reoccurring Situations associated with a team in the specified time range.
- [getServiceSituationPerTeamStats](#): Returns the number of Situations impacting each service for a team.
- [getSeveritySituationPerTeamStats](#): Returns the number of Situations by severity per team in the specified time range.
- [getStatusSituationPerTeamStats](#): Returns the number of Situations by status for a team in the specified time range.
- [getTeamSituationStats](#): Returns the number of active Situations assigned to a team in the specified time range.
- [getTopServiceSituationStats](#): Returns the number of active Situations impacting a top service in the specified time range.

User Endpoints

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

- [getAlertsMarkedPRCPerUserStats](#): Returns the total number of alerts marked with probable root cause (PRC) feedback by each user.
- [getAcknowledgedSituationsPerUserStats](#): Returns the number of Situations acknowledged by a specific user or users in the specified time range.
- [getAssignedSituationsPerUserStats](#): Returns the number of Situations assigned to a specific user or users in the specified time range.
- [getChatOpsToolExecutedPerUserStats](#): Returns the number of ChatOps tools executed by a user each hour in the specified time range.
- [getClosedSituationsPerUserStats](#): Returns the number of Situations that a user has closed each hour in the specified time range.
- [getCommentCountPerUserStats](#): Returns the number of comments left by a user or users in the specified time range.
- [getInvitationsReceivedPerUserStats](#): Returns the number of Situation invitations received for a given user each hour in the specified time range.
- [getMTTAPerUserStats](#): Returns the mean time it takes a user to acknowledge a Situation in the specified time range.
- [getMTTRPerUserStats](#): Returns the mean time it takes a user to resolve a Situation in the specified time range.
- [getOpenSituationsPerUserStats](#): Returns the number of open Situations assigned to a user at each data point.
- [getRatedSituationsPerUserStats](#): Returns the number of Situations rated by a user in the specified time range.
- [getReassignedSituationsPerUserStats](#): Returns the number of Situations reassigned by a user in the specified time range.
- [getResolvedSituationsPerUserStats](#): Returns the number of Situations resolved by a user in the specified time range.
- [getViewedSituationsPerUserStats](#): Returns the number of Situations a user has viewed in the specified time range.
- [getWorkedSituationsPerUserStats](#): Returns the number of Situations a user has worked on in the specified time range.

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](#): Returns the number of alerts that belong to new Situations in the specified time range.

- [getMTTASStats](#): Returns the Mean Time To Acknowledge (MTTA) Situations in the specified time range.
- [getMTTDStats](#): Returns the Mean Time To Detect (MTTD) Situations in the specified time range.
- [getMTTRStats](#): Returns the Mean Time To Resolve (MTTR) for Situations in the specified time range.
- [getNewAlertsStats](#): Returns the number of new alerts in the specified time range.
- [getNewAlertsPerSituationsStats](#): Returns the percentage of noise reduction from alerts-to-Situations clustering in the specified time range.
- [getNewEventsPerAlertsStats](#): Returns the percentage of noise reduction from events-to-alerts aggregation and deduplication in the specified time range.
- [getNewEventsPerSituationsStats](#): Returns the percentage of noise reduction from events-to-Situations aggregation, deduplication, and clustering in the specified time range.
- [getNewSituationsStats](#): Returns the number of new Situations created in the specified time range.
- [getReassignedSituationStats](#): Returns the number of Situations reassigned in the specified time range.
- [getReoccurringSituationStats](#): Returns the percentage of reoccurring situations in the specified time range.
- [getServiceSituationStats](#): Returns the number of active Situations impacting a service in the specified time range.
- [getSeveritySituationStats](#): Returns the number of Situations by severity in the specified time range.
- [getStats](#): Returns all available Stats API endpoints along with their description and request parameters.
- [getStatusSituationStats](#): Returns the number of Situations by status.
- [getSystemSituationStats](#): Returns the number of active Situations in the specified time range.
- [getTopServiceSituationStats](#): Returns the number of active Situations impacting a top service in the specified time range.

Team Endpoints

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

- [getCommentCountPerTeamStats](#): Returns the total number of comments each hour for a specific team or teams in the specified time range.
- [getMTTAPerTeamStats](#): Returns the mean time to acknowledge (MTTA) a Situation per team in the specified time range.
- [getMTTRPerTeamStats](#): Returns the mean time to resolve (MTTR) a Situation per team in the specified time range.
- [getReassignedSituationsPerTeamStats](#): Returns the number of reassigned Situations associated with a team or multiple teams in the specified time range.

- [getReoccurringSituationPerTeamStats](#): Returns the number of reoccurring Situations associated with a team in the specified time range.
- [getServiceSituationPerTeamStats](#): Returns the number of Situations impacting each service for a team.
- [getSeveritySituationPerTeamStats](#): Returns the number of Situations by severity per team in the specified time range.
- [getStatusSituationPerTeamStats](#): Returns the number of Situations by status for a team in the specified time range.
- [getTeamSituationStats](#): Returns the number of active Situations assigned to a team in the specified time range.
- [getTopServiceSituationStats](#): Returns the number of active Situations impacting a top service in the specified time range.

User Endpoints

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

- [getAlertsMarkedPRCPerUserStats](#): Returns the total number of alerts marked with probable root cause (PRC) feedback by each user.
- [getAcknowledgedSituationsPerUserStats](#): Returns the number of Situations acknowledged by a specific user or users in the specified time range.
- [getAssignedSituationsPerUserStats](#): Returns the number of Situations assigned to a specific user or users in the specified time range.
- [getChatOpsToolExecutedPerUserStats](#): Returns the number of ChatOps tools executed by a user each hour in the specified time range.
- [getClosedSituationsPerUserStats](#): Returns the number of Situations that a user has closed each hour in the specified time range.
- [getCommentCountPerUserStats](#): Returns the number of comments left by a user or users in the specified time range.
- [getInvitationsReceivedPerUserStats](#): Returns the number of Situation invitations received for a given user each hour in the specified time range.
- [getMTTAPerUserStats](#): Returns the mean time it takes a user to acknowledge a Situation in the specified time range.
- [getMTTRPerUserStats](#): Returns the mean time it takes a user to resolve a Situation in the specified time range.
- [getOpenSituationsPerUserStats](#): Returns the number of open Situations assigned to a user at each data point.
- [getRatedSituationsPerUserStats](#): Returns the number of Situations rated by a user in the specified time range.
- [getReassignedSituationsPerUserStats](#): Returns the number of Situations reassigned by a user in the specified time range.
- [getResolvedSituationsPerUserStats](#): Returns the number of Situations resolved by a user in the specified time range.

- [getViewedSituationsPerUserStats](#): Returns the number of Situations a user has viewed in the specified time range.
- [getWorkedSituationsPerUserStats](#): Returns the number of Situations a user has worked on in the specified time range.

getAlertsInNewSituationsStats

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

Back to [Stats API](#).

Request arguments

Endpoint **getAlertsInNewSituationsStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getAlertsInNewSituationsStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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 [data point, timestamp]: Data point: Number of alerts. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getAlertsInNewSituationsStats**:

Request example

A cURL GET request for 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 at 13:23pm on Wednesday 19th September 2018:

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

getAcknowledgedSituationsPerUserStats

A GET request that returns the number of Situations acknowledged by a specific user or users within a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getAcknowledgedSituationsPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getAcknowledgedSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

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

target	String	"Acknowledged Situations (full name)"
datapoints	Number Array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <p>Data point: Number of Situations acknowledged by the user.</p> <p>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).</p> <p>Less than 1 week: Returns the number of Situations acknowledged each hour in the time period.</p> <p>1 week to 1 month: Returns the number of Situations acknowledged each day in the time period.</p> <p>1 month to 1 year: Returns the number of Situations acknowledged each week in the time period.</p> <p>More than 1 year: Returns the number of Situations acknowledged each month in the time period.</p>

Examples

The following examples demonstrate typical use of endpoint **getAcknowledgedSituationsPerUserStats**:

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)"
}]
```

getAlertsMarkedPRCPerUserStats

A GET request that returns the total number of alerts marked with probable root cause (PRC) feedback by each user.

Back to [Stats API](#).

Request arguments

Endpoint **getAlertsMarkedPRCPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getAlertsMarkedPRCPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of alerts marked with PRC feedback by the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). 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. 1 month to 1 year: Returns the number of alerts marked with PRC feedback each week in the time period. More than 1 year: Returns the number of alerts marked with PRC feedback each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getAlertsMarkedPRCPerUserStats**:

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 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)"
}]
```

getAssignedSituationsPerUserStats

A GET request that returns the number of Situations assigned to a specific user or users within a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getAssignedSituationsPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getAssignedSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations assigned to the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getAssignedSituationsPerUserStats**:

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.

Back to [Stats API](#).

Request arguments

Endpoint **getChatOpsToolExecutedPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getChatOpsToolExecutedPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of ChatOps tools executed by the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of ChatOps tools executed each

		<p>hour in the time period.</p> <p>1 week to 1 month: Returns the number of ChatOps tools executed each day in the time period.</p> <p>1 month to 1 year: Returns the number of ChatOps tools executed each week in the time period.</p> <p>More than 1 year: Returns the number of ChatOps tools executed each month in the time period.</p>
--	--	---

Examples

The following examples demonstrate typical use of endpoint

getChatOpsToolExecutedPerUserStats:

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.

Back to [Stats API](#).

Request arguments

Endpoint **getClosedSituationsPerUserStats** takes the following request arguments.

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

auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getClosedSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations closed by the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of Situations closed each hour in the time period. 1 week to 1 month: Returns the number of Situations closed each day in the time period. 1 month to 1 year: Returns the number of Situations closed each week in the time period. More than 1 year: Returns the number of Situations closed each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getClosedSituationsPerUserStats**:

Request example

A cURL request to return the number of Situations closed by user 5 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)"
}]
```

getCommentCountPerTeamStats

A GET request that returns the total number of comments each hour for a specific team or teams in a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getCommentCountPerTeamStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getCommentCountPerTeamStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of comments. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getCommentCountPerTeamStats**:

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" }
        ]
      }
    ]
  }
]
```

getCommentCountPerUserStats

A GET request that returns the number of comments left by a user or users within a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getCommentCountPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getCommentCountPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of comments. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of comments each hour in the time period. 1 week to 1 month: Returns the number of comments each day in the time period. 1 month to 1 year: Returns the number of comments each week in the time period. More than 1 year: Returns the number of comments each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getCommentCountPerUserStats**:

Request example

A cURL request to retrieve the total number of comments made by users 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/getCommentCountPerUserStats" --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.

Back to [Stats API](#).

Request arguments

Endpoint **getInvitationsReceivedPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getInvitationsReceivedPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Invitations Received (full name)"
datapoints	Number Array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <p>Data point: Invitations received by the user.</p> <p>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).</p> <p>Less than 1 week: Returns the number of Situation invitations each hour in the time period.</p> <p>1 week to 1 month: Returns the number of Situation invitations each day in the time period.</p> <p>1 month to 1 year: Returns the number of Situation invitations each week in the time period.</p> <p>More than 1 year: Returns the number of Situation invitations each month in the time period.</p>

Examples

The following examples demonstrate typical use of endpoint

getInvitationsReceivedPerUserStats:

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)"
}
]

```

getMTTASStats

A GET request that returns the Mean Time To Acknowledge (MTTA) Situations in the specified time range.

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 Moogsoft AIOps.

Back to [Stats API](#).

Request arguments

Endpoint **getMTTASStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getMTTASStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: MTTA (seconds) for that bucket. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getMTTASStats**:

Request example

A cURL command to return the MTTA for Moogsoft AIOps 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/getMTTASStats" --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)"
}]
```

getMTTAPerTeamStats

A GET request that returns the mean time to acknowledge (MTTA) a Situation per team in the specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getMTTAPerTeamStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getMTTAPerTeamStats** returns the following response:

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

Code	definitions for more information.
------	---

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]: Data point:MTTA (seconds) for that bucket. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getMTTAPerTeamStats**:

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 13.14pm 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)"
}]
```

getMTTAPerUserStats

A GET request that returns the mean time it takes a user to acknowledge a Situation within a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getMTTAPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getMTTAPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point:MTTA (seconds) for that bucket. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of MTTA each hour in the time period. 1 week to 1 month: Returns the number of MTTA each day in the time period. 1 month to 1 year: Returns the number of MTTA each week in the time period. More than 1 year: Returns the number of MTTA each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getMTTAPerUserStats**:

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:

Cisco Systems, Inc. www.cisco.com

```
[{
  "datapoints": [
    [221,1537857295000],
    [960,1537860895000],
    [901,1537864495000],
    [1196,1537868095000],
    [671,1537871695000],
    [1241,1537875295000],
    [556,1537878895000]
  ],
  "target": "Mean Time to Acknowledge (MTTA)(Robert Richards)"
}]
```

getMTTDStats

A GET request that returns the Mean Time To Detect (MTTD) Situations in the specified time range.

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.

Back to [Stats API](#).

Request arguments

Endpoint **getMTTDStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getMTTDStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: MTTD (seconds) for that bucket Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getMTTDStats**:

Request example

A cURL request to retrieve the MTTR for Moogsoft AIOps 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/getMTTRStats" --data-urlencode 'from=1513508950' --data-urlencode 'to=1514113750'
```

Response example

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

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

getMTTRStats

A GET request that returns the Mean Time To Resolve (MTTR) for Situations in the specified time range.

The TTR for a Situation is the duration from the first event in the Situation to the time when a user resolved the Situation.

Back to [Stats API](#).

Request arguments

Endpoint **getMTTRStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getMTTRStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: MTTR (seconds) for that bucket

		Timestamp: Calculation time (Unix epoch timestamp in milliseconds).
--	--	---

Examples

The following examples demonstrate typical use of endpoint **getMTTRStats**:

Request example

A cURL request to retrieve the MTTR for Moogsoft AIOps from 11.30am on Sunday, September 24th 2017 until 11.30am on Sunday, September 24th 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)"
}]
```

getMTTRPerTeamStats

A GET request that returns the mean time to resolve (MTTR) a Situation per team for a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getMTTRPerTeamStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getMTTRPerTeamStats** returns the following response:

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

Code	definitions for more information.
------	---

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]: Data point: MTTR (seconds) for that bucket. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getMTTRPerTeamStats**:

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)"
}]
```

getMTTRPerUserStats

A GET request that returns the mean time it takes a user to resolve a Situation within a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getMTTRPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getMTTRPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: MTTR (seconds) for that bucket. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of MTTR each hour in the time period. 1 week to 1 month: Returns the number of MTTR each day in the time period. 1 month to 1 year: Returns the number of MTTR each week in the time period. More than 1 year: Returns the number of MTTR each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getMTTRPerUserStats**:

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)"
}]
```

getNewAlertsPerSituationsStats

A GET request that returns the percentage of noise reduction from alerts-to-Situations clustering in the specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getNewAlertsPerSituationsStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getNewAlertsPerSituationsStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Percentage noise reduction (alert to Situation reduction). Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint

getNewAlertsPerSituationsStats:

Request example

Example 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

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

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

getNewAlertsStats

A GET request that returns the number of new alerts in the specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getNewAlertsStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getNewAlertsStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of alerts Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getNewAlertsStats**:

Request example

Example 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

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

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

getNewEventsPerAlertsStats

A GET request that returns the percentage of noise reduction from events-to-alerts aggregation and deduplication in the specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getNewEventsPerAlertsStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getNewEventsPerAlertsStats** returns the following response:

Type	Description
------	-------------

HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.
-----------	---

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	" New Events per Alerts"
datapoints	Number Array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <p>Data point: Percentage noise reduction (event to alert reduction).</p> <p>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).</p> <p>Less than 1 week: Returns the number of percentage noise reduction each hour in the time period.</p> <p>1 week to 1 month: Returns the number of percentage noise reduction each day in the time period.</p> <p>1 month to 1 year: Returns the number of percentage noise reduction each week in the time period.</p> <p>More than 1 year: Returns the number of percentage noise reduction each month in the time period.</p>

Examples

The following examples demonstrate typical use of endpoint **getNewEventsPerAlertsStats**:

Request example

A cURL request that retrieves that event to alert noise reduction in Moogsoft AIOps 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 of noise reduction from events-to-Situations aggregation, deduplication, and clustering in the specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getNewEventsPerSituationsStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getNewEventsPerSituationsStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"New Events per Situation"
datapoints	Number Array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <p>Data point: Percentage noise reduction (event to Situation reduction).</p> <p>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).</p> <p>[Delete all except the appropriate Time Period box or complete the custom list if not supplied. Delete this para!]</p>

Examples

The following examples demonstrate typical use of endpoint **getNewEventsPerSituationsStats**:

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"
}
]

```

getNewEventsPerSituationsStats

A GET request that returns the percentage of noise reduction from events-to-Situations aggregation, deduplication, and clustering in the specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getNewEventsPerSituationsStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getNewEventsPerSituationsStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	" New Events per Situation"
datapoints	Number Array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <p>Data point: Percentage noise reduction (event to Situation reduction).</p> <p>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).</p> <p>[Delete all except the appropriate Time Period box or complete the custom list if not supplied. Delete this para!]</p>

Examples

The following examples demonstrate typical use of endpoint **getNewEventsPerSituationsStats**:

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 in the specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getNewSituationsStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getNewSituationsStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

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

target	String	"New Situations"
datapoints	Number Array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <p>Data point: Number of new Situations.</p> <p>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).</p> <p>Less than 1 week: Returns the number of new Situations each hour in the time period.</p> <p>1 week to 1 month: Returns the number of new Situations each day in the time period.</p> <p>1 month to 1 year: Returns the number of new Situations each week in the time period.</p> <p>More than 1 year: Returns the number of new Situations each month in the time period.</p>

Examples

The following examples demonstrate typical use of endpoint **getNewSituationsStats**:

Request example

Example 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

Example response returning 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"
  }
]
```

getOpenSituationsPerUserStats

A GET request that returns the number of open Situations assigned to a user at each data point.

Back to [Stats API](#).

Request arguments

Endpoint **getOpenSituationsPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getOpenSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of open Situations assigned to the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of Situations assigned each hour in the time period. 1 week to 1 month: Returns the number of Situations assigned each day in the time period. 1 month to 1 year: Returns the number of Situations assigned each week in the time period. More than 1 year: Returns the number of Situations assigned each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getOpenSituationsPerUserStats**:

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.

Back to [Stats API](#).

Request arguments

Endpoint **getRatedSituationsPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.

aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.
--------------------	--------	---

Response

Endpoint **getRatedSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations rated by the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of Situations rated each hour in the time period. 1 week to 1 month: Returns the number of Situations rated each day in the time period. 1 month to 1 year: Returns the number of Situations rated each week in the time period. More than 1 year: Returns the number of Situations rated each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getRatedSituationsPerUserStats**:

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)"
}]

```

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.

Back to [Stats API](#).

Request arguments

Endpoint **getReassignedSituationsPerTeamStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getReassignedSituationsPerTeamStats** returns the following response:

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

Code	definitions for more information.
------	---

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]: Data point: Number of reassigned Situations. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getReassignedSituationsPerTeamStats**:

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"
}]
```

getReassignedSituationsPerUserStats

A GET request that returns the number of Situations reassigned by a user within a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getReassignedSituationsPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getReassignedSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations reassigned by the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of Situations reassigned each hour in the time period. 1 week to 1 month: Returns the number of Situations reassigned each day in the time period. 1 month to 1 year: Returns the number of Situations reassigned each week in the time period. More than 1 year: Returns the number of Situations reassigned each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getReassignedSituationsPerUserStats**:

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)"
}]
```

getReassignedSituationStats

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

Back to [Stats API](#).

Request arguments

Endpoint **getReassignedSituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getReassignedSituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of reassigned Situations. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getReassignedSituationStats**:

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"
}]
```

getReoccurringSituationPerTeamStats

A GET request that returns the number of reoccurring Situations associated with a team for a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getReoccurringSituationPerTeamStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getReoccurringSituationPerTeamStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of reoccurring Situations. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getReoccurringSituationPerTeamStats**:

Request example

A cURL request to retrieve the number of reoccurring 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 reoccurring Situations at the time the request was sent:

```
[{"datapoints":[[4.0,1538044321144]],"target":"Reoccurring situations"}]
```

getReoccurringSituationStats

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

Back to [Stats API](#).

Request arguments

Endpoint **getReoccurringSituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getReoccurringSituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of reoccurring Situations. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getReoccurringSituationStats**:

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"
}]
```

getResolvedSituationsPerUserStats

A GET request that returns the number of Situations resolved by a user within a given time range.

Back to [Stats API](#).

Request arguments

Endpoint **getResolvedSituationsPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getResolvedSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations resolved by the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds). Less than 1 week: Returns the number of Situations resolved each hour in the time period. 1 week to 1 month: Returns the number of Situations resolved each day in the time period. 1 month to 1 year: Returns the number of Situations resolved each week in the time period. More than 1 year: Returns the number of Situations resolved each month in the time period.

Examples

The following examples demonstrate typical use of endpoint **getResolvedSituationsPerUserStats**:

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)"
}]
```

getServiceSituationPerTeamStats

A GET request that returns the number of Situations impacting each service for a team.

Back to [Stats API](#).

Request arguments

Endpoint **getServiceSituationPerTeamStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getServiceSituationPerTeamStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations impacting services. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint

getServiceSituationPerTeamStats:

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"}]
```

getServiceSituationStats

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

Back to [Stats API](#).

Request arguments

Endpoint **getServiceSituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
services	Array	An array of services IDs. If no services 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getServiceSituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations impacting services. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getServiceSituationStats** :

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"
}]
```

getSeveritySituationPerTeamStats

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

Back to [Stats API](#).

Request arguments

Endpoint **getSeveritySituationPerTeamStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getSeveritySituationPerTeamStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations per severity. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint

getSeveritySituationPerTeamStats:

Request example

A cURL request to retrieve the number of clear Situations for the Cloud DevOps team between between 12pm on Thursday, 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"
}]
```

getSeveritySituationStats

A GET request that returns the number of Situations by severity in the specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getSeveritySituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
services	Array	An array of services IDs. If no services 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getSeveritySituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations per severity. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getSeveritySituationStats**:

Request 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'
```

Response 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.

Back to [Stats API](#).

Request arguments

Name	Type	Required	Description
auth_token	String	Yes	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.

Endpoint **getStats** takes no other arguments because it returns data on all available Stats API endpoints.

Response

Endpoint **getStats** returns the following response:

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

Code	definitions for more information.
------	---

Examples

The following examples demonstrate typical use of endpoint **getStats**:

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": "getMTTASStats",
    "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"
                }
            ],
            "required":false
        }
    }
}

```

```

        {
            "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",
        "value": "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
        }
    }
}
]

```

getStatusSituationPerTeamStats

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

Cisco Systems, Inc. www.cisco.com

Back to [Stats API](#).

Request arguments

Endpoint **getStatusSituationPerTeamStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getStatusSituationPerTeamStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations for each status. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getStatusSituationPerTeamStats**:

Request example

A cURL request 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"
  }
]
```

getStatusSituationStats

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

Back to [Stats API](#).

Request arguments

Endpoint **getStatusSituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getStatusSituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The status name.
datapoints	Number Array	An array of data points. Each data point is an array in the format [data point, timestamp]: Data point: Number of Situations for each status Timestamp: Calculation time (Unix epoch timestamp in milliseconds). [Delete all except the appropriate Time Period box or complete the custom list if not supplied. Delete this para!]

Examples

The following examples demonstrate typical use of endpoint **getStatusSituationStats**:

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

Example response returning 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 specified time range.

Back to [Stats API](#).

Request arguments

Endpoint **getSystemSituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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

Endpoint **getSystemSituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"System"
datapoi	Num ber	An array of data points. Each data point is an array in the format [data point,

nts	Array	timestamp]: Data point: Number of Situations for each Status Timestamp: Calculation time (Unix epoch timestamp in milliseconds). [Delete all except the appropriate Time Period box or complete the custom list if not supplied. Delete this para!]
------------	-------	--

Examples

The following examples demonstrate typical use of endpoint **getSystemSituationStats**:

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"
}]
```

getTeamSituationStats

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

Back to [Stats API](#).

Request arguments

Endpoint **getTeamSituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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. 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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getTeamSituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations for each status. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getTeamSituationStats**:

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.

Back to [Stats API](#).

Request arguments

Endpoint **getTopTeamSituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getTopTeamSituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations for each status. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getTopTeamSituationStats**:

Request example

A cURL request to retrieve the number of Situations impacting top teams between 6am and 12pm on Wednesday, 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 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"
}]
```

getTopServiceSituationStats

A GET request that returns the number of active Situations impacting a top service in the specified time range. Top services are the services that have the most situations impacting them.

Back to [Stats API](#).

Request arguments

Endpoint **getTopServiceSituationStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
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.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getTopServiceSituationStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations for each status. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

Examples

The following examples demonstrate typical use of endpoint **getTopServiceSituationStats**:

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"
}]

```

getViewedSituationsPerUserStats

A GET request that returns the number of Situations a user has viewed within a given time range. Moogsoft AIOps considers a user to have viewed a Situation if they opened the Situation Room.

Back to [Stats API](#).

Request arguments

Endpoint **getViewedSituationsPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getViewedSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

Successful requests return a JSON object containing the following:

Cisco Systems, Inc. www.cisco.com

Name	Type	Description
target	String	"Viewed Situations (full name)"
datapoints	Number Array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <p>Data point: Number of Situations viewed by the user.</p> <p>Timestamp: Calculation time (Unix epoch timestamp in milliseconds).</p> <p>Less than 1 week: Returns the number of Situations viewed each hour in the time period.</p> <p>1 week to 1 month: Returns the number of Situations viewed each day in the time period.</p> <p>1 month to 1 year: Returns the number of Situations viewed each week in the time period.</p> <p>More than 1 year: Returns the number of Situations viewed each month in the time period.</p>

Examples

The following examples demonstrate typical use of endpoint **getViewedSituationsPerUserStats**:

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 Cisco Crosswork Situation Manager considers a user to have worked on a Situation if the user has:

1. Been assigned a Situation.
2. Been invited to a Situation.
3. Left a comment on a Situation.
4. Closed a Situation.
5. Resolved a Situation.
6. Executed a ChatOps tool on a Situation.
7. Rated a Situation.
8. Added PRC data to alerts in a Situation.

Back to [Stats API](#).

Request arguments

Endpoint **getWorkedSituationsPerUserStats** takes the following request arguments.

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request. See the authenticate endpoint for more information.
users	Array	An array of user IDs. If no users 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. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	Set to: accumulate : Gradually adds data points together over time. none : No aggregation of data points.

Response

Endpoint **getWorkedSituationsPerUserStats** returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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]: Data point: Number of Situations worked on by the user. Timestamp: Calculation time (Unix epoch timestamp in milliseconds).

		<p>Less than 1 week: Returns the number of worked Situations each hour in the time period.</p> <p>1 week to 1 month: Returns the number of worked Situations each day in the time period.</p> <p>1 month to 1 year: Returns the number of worked Situations each week in the time period.</p> <p>More than 1 year: Returns the number of worked Situations each month in the time period.</p>
--	--	---

Examples

The following examples demonstrate typical use of endpoint **getWorkedSituationsPerUserStats**:

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)"
}]
```

Integrations API

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

Contact Cisco Support if you experience difficulties or need further guidance.

Endpoints

See [Integrations API Endpoint Reference](#) for details of all the Integrations API endpoints.

API definition

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

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

Examples:

`https://localhost/graze/v1/integrations/`

`https://localhost/graze/v1/integrations/{integrationId}`

`https://localhost/graze/v1/integrations/{integrationId}/status`

Authentication

In order to use the Integrations API, you must have the `manage_integrations` permission. See [/document/preview/35141#UUID1b6353d943aee6691d57d631ed647220](#) for more information. Role Permissions

All requests require a basic authentication header.

Integrations API Endpoint Reference

This is a reference list for the Integrations API endpoints. Follow the links to see the details of each endpoint.

All of the endpoints use basic authorization.

Brokers

The following endpoints relate to brokers:

- [/broker-profiles](#)Create a Broker Profile

Integrations

The following endpoints relate to integration management:

1. [/integrations](#)
2. [/integrations/{integrationId}](#)
3. [/integrations/{integrationId}/status](#)

/integrations

The **/integrations** endpoint allows you to create new integrations.

To read and update existing integrations see [/integrations/{integrationId}](#).

Back to [Integrations API Endpoint Reference](#).

POST

Creates an integration's configuration.

Request arguments

The POST request takes the following request arguments:

Name	Type	Required	Description
type_id	String	Yes	Type of integration to add, for example Webhook
inputs	String	Yes	The key and value of inputs to substitute into the integration's configuration. This can include username, password, URL to poll, and timing intervals.
name	String	Yes	Name of the integration to add, for example Webhook1
version	String	Yes	Version of the integration to use. For validation purposes, this must be the most recent version.

Response

The POST request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
readonly	List	Read-only details about the integration. This can include the webhook URL of the integration and authorisation details.
type_id	String	Type of integration you have created.
inputs	String	Username (value) and password (key) you have configured to authenticate with the integration.
name	String	Name of the integration you have created.
id	Integer	ID of the integration you have created.
version	String	Version of the integration you have created.
config	Object	The integration's configuration.

Examples

The following examples demonstrate typical use making a POST request to the endpoint **integrations**:

Request example

Example cURL POST request to create a Webhook integration:

```
curl -X POST \
https://example.com/integrations/api/v1/integrations \
-u John.Doe:MyPassword \
-d '{
  "type_id": "Webhook",
  "inputs": [
    {
      "name": "username",
      "value": "<username>"
    },
    {
      "name": "password",
      "value": "<integrationpassword>"
    }
  ],
  "name": "Webhook1",
  "version": "1.14"
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

Example response returning the new Webhook integration's configuration:

```
{
  "id": 4,
  "name": "Webhook2",
  "type_id": "Webhook",
  "version": "1.14",
  "config": {
    "category": "monitoring",
    "description": "A webhook integration to allow events to be sent
via generic REST and processed by Moogsoft AIOps.",
    "display_name": "Webhook",
    "type_id": "Webhook",
    "version": "1.14",
    "config": {
      "monitor": {
        "name": "Webhook Lam Monitor",
        "class": "CRestMonitor",
        "port": "$config#port()",
        "authentication_type": "basic_auth_static",
        "basic_auth_static": {
          "username": "John.Doe",
          "password": "Password123"
        }
      },
      "use_ssl": false,
      "accept_all_json": true,
      "lists_contain_multiple_events": true,
      "num_threads": 5,
      "rest_response_mode": "on_receipt",
      "rpc_response_timeout": 20
    }
  },
}
```

```
"constants": {
  "severity": {
    "CLEAR": 0,
    "INDETERMINATE": 1,
    "WARNING": 2,
    "MINOR": 3,
    "MAJOR": 4,
    "CRITICAL": 5,
    "0": 0,
    "1": 1,
    "2": 2,
    "3": 3,
    "4": 4,
    "5": 5,
    "moog_lookup_default": 1
  }
},
"conversions": {
  "sevConverter": {
    "input": "STRING",
    "output": "INTEGER",
    "lookup": "severity"
  },
  "stringToInt": {
    "input": "STRING",
    "output": "INTEGER"
  }
},
"filter": {
  "presend": "WebhookLam-SolutionPak.js",
  "modules": [],
  "dependencies": {
    "lambot": [
      "LamBot.js",
      "WebhookLam-SolutionPak.js"
    ],
    "contrib": []
  }
},
"mapping": {
  "lambotOverridden": [],
  "catchAll": "overflow",
  "rules": [
    {
      "name": "signature",
      "rule": "$source::$type"
    },
    {
      "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"
    }
    ]
}
},
"ha_profile": "active_active"
},
"inputs": [
    {
        "key": "username",
        "value": "password"
    }
],
"readonly": [
    {
        "name": "url",
        "description": "URL:",
        "value":
"https://example.com/integrations/api/v1/events/webhook2"
    },
    {

```

```

        "name": "userid",
        "description": "User ID:",
        "value": "<username>"
    },
    {
        "name": "readonly_password",
        "description": "Password:",
        "value": "<password>"
    },
    {
        "name": "auth",
        "description": "Base64 Encoded Auth:",
        "value": "Basic YWRtaW46"
    }
  ]
}

```

/integrations/{integrationId}

The **/integrations/{integrationId}** endpoint allows you to read and update existing integrations.

To create a new integration see [/integrations](#).

Back to [Integrations API Endpoint Reference](#).

GET

Retrieves a specific integration's configuration.

Request arguments

The GET request takes the following request argument:

Name	Type	Required	Description
integrationId	Integer	Yes	ID of the integration to retrieve. You can obtain an integration's ID by executing a GET request to /integrations .

There are no other arguments because this endpoint returns data based on the integration ID alone.

Response

The GET request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
readonly	List	Read-only details about the integration. This can include the webhook URL of the integration and authorization details.
type_id	String	Type of integration.
inputs	String	Username (value) and password (key) you have configured to authenticate with the integration.

name	String	Name of the integration.
id	Integer	ID of the integration.
version	String	Version of the integration. For validation purposes, this must be the most recent version.
config	Object	Integration's configuration.

Examples

The following examples demonstrate making a GET request to the endpoint **integrations/{integrationId}**:

Request example

Example cURL request for details of the integration with the ID **"3"**:

```
curl \
https://example.com/integrations/api/v1/integrations/3 \
-u John.Doe:MyPassword \
```

Response example

A successful request returns the HTTP code 200 and no response text.

Example response returning the integration's details:

```
{
  "id": 3,
  "name": "DynatraceAPMPolling1",
  "type_id": "dynatrace_apm_lam",
  "version": "2.3",
  "config": {
    "config": {
      "filter": {
        "modules": [],
        "presend": "DynatraceApmLam.js",
        "dependencies": {
          "lambot": [
            "LamBot.js",
            "DynatraceApmLam.js"
          ],
          "contrib": []
        }
      },
      "mapping": {
        "rules": [
          {
            "name": "signature",
            "rule": "$systemprofile :: $rule"
          },
          {
            "name": "source_id",
            "rule": "Dynatrace APM"
          },
          {
            "name": "external_id",
```

```
        "rule": "$id"
      },
      {
        "name": "manager",
        "rule": "Dynatrace Apm"
      },
      {
        "name": "source",
        "rule": "$source"
      },
      {
        "name": "class",
        "rule": "$rule"
      },
      {
        "name": "agent",
        "rule": "$LamInstanceName"
      },
      {
        "name": "agent_location",
        "rule": "$LamInstanceName"
      },
      {
        "name": "type",
        "rule": "$state"
      },
      {
        "name": "severity",
        "rule": "$severity",
        "conversion": "sevConverter"
      },
      {
        "name": "description",
        "rule": "$message"
      },
      {
        "name": "agent_time",
        "rule": "$start",
        "conversion": "timeConverter"
      }
    ],
    "catchAll": "overflow",
    "lambotOverridden": [
      "custom_info.overflow",
      "source"
    ]
  },
  "monitor": {
    "name": "DynatraceApm Lam Monitor",
    "class": "CDynatraceApmMonitor",
    "targets": {
      "target1": {
        "url": "https://localhost:8021",
        "filter": {
          "state": "InProgress",
          "profileName": "nam",
```

```

        "incidentRule": "rul"
      },
      "timeout": 120,
      "password": "def",
      "username": "abc",
      "disable_certificate_validation": true
    }
  },
  "max_retries": -1,
  "retry_interval": 60,
  "request_interval": 60
},
"constants": {
  "severity": {
    "severe": 5,
    "warning": 2,
    "informational": 1
  }
},
"conversions": {
  "stringToInt": {
    "input": "STRING",
    "output": "INTEGER"
  },
  "sevConverter": {
    "input": "STRING",
    "lookup": "severity",
    "output": "INTEGER"
  },
  "timeConverter": {
    "input": "STRING",
    "output": "INTEGER",
    "timeFormat": "yyyy-MM-dd'T'HH:mm:ss.SSS"
  }
}
},
"moolets": [],
"type_id": "dynatrace_apm_lam",
"version": "2.3",
"category": "monitoring",
"ha_profile": "active_passive",
"description": "An integration which enables Moogsoft AIOps to
ingest events from Dynatrace APM.",
"display_name": "Dynatrace APM (Polling)"
},
"inputs": [
  {
    "key": "targets",
    "value": [
      {
        "url": "https://localhost:8021",
        "filter": {
          "state": "InProgress",
          "profile_name": "nam",
          "incident_rule": "rul"
        }
      }
    ]
  }
]

```

```

        "password": "def",
        "username": "abc"
      }
    ],
    {
      "key": "timing",
      "value": {
        "timeout": 120,
        "retry_interval": 60,
        "request_interval": 60
      }
    }
  ],
  "readonly": null
}

```

PUT

Updates an integration's configuration. Integrations with the ID you specify are unavailable during the update. When the update completes, they automatically resume.

Request arguments

The PUT request takes the following request argument:

Name	Type	Required	Description
integrationId	Integer	Yes	ID of the integration to update.
type_id	String	Yes	Type of integration to add, for example Webhook .
inputs	String	Yes	The key and value of inputs to substitute into the integration's configuration. This can include username, password, URL to poll, and timing intervals.
name	String	Yes	Name of the integration to add, for example Webhook1 .
version	String	Yes	Version of the integration to use. For validation purposes, this must be the most recent version.

Response

The PUT request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
readonly	List	Read-only details about the integration. This can include the webhook URL of the integration and authorisation details.
type_id	String	Type of integration you have updated.
inputs	String	Username (value) and password (key) you have configured to authenticate with the integration.

name	String	Name of the integration you have updated.
id	Integer	ID of the integration you have updated.
version	String	Version of the integration you have updated.
config	Object	The integration's configuration.

Examples

The following examples demonstrate making a PUT request to the endpoint **integrations/{integrationId}**:

Request example

Example cURL PUT request to update the **value** and **password** parameters for a Webhook integration. In this example, the Webhook's ID is **2**:

```
curl -X PUT \
https://example.com/integrations/api/v1/integrations/2 \
-u John.Doe:MyPassword \
-d '{
  "type_id": "Webhook",
  "inputs": [
    {
      "name": "username",
      "value": "Jane.Doe"
    },
    {
      "name": "password",
      "value": "Password123"
    }
  ],
  "name": "Webhook1",
  "version": "1.14"
}'
```

Response example

A successful request returns the HTTP code 200 and no response text.

Example response returning the updated integration's details:

```
{
  "id": 2,
  "name": "Webhook2",
  "type_id": "Webhook",
  "version": "1.14",
  "config": {
    "category": "monitoring",
    "description": "A webhook integration to allow events to be sent
via generic REST and processed by Moogsoft AIOps.",
    "display_name": "Webhook",
    "type_id": "Webhook",
    "version": "1.14",
    "config": {
      "monitor": {
        "name": "Webhook Lam Monitor",
```

```
    "class": "CRestMonitor",
    "port": "$config#port()",
    "authentication_type": "basic_auth_static",
    "basic_auth_static": {
      "username": "<username>",
      "password": "<password>"
    },
    "use_ssl": false,
    "accept_all_json": true,
    "lists_contain_multiple_events": true,
    "num_threads": 5,
    "rest_response_mode": "on_receipt",
    "rpc_response_timeout": 20
  },
  "constants": {
    "severity": {
      "CLEAR": 0,
      "INDETERMINATE": 1,
      "WARNING": 2,
      "MINOR": 3,
      "MAJOR": 4,
      "CRITICAL": 5,
      "0": 0,
      "1": 1,
      "2": 2,
      "3": 3,
      "4": 4,
      "5": 5,
      "moog_lookup_default": 1
    }
  },
  "conversions": {
    "sevConverter": {
      "input": "STRING",
      "output": "INTEGER",
      "lookup": "severity"
    },
    "stringToInt": {
      "input": "STRING",
      "output": "INTEGER"
    }
  },
  "filter": {
    "presend": "WebhookLam-SolutionPak.js",
    "modules": [],
    "dependencies": {
      "lambot": [
        "LamBot.js",
        "WebhookLam-SolutionPak.js"
      ],
      "contrib": []
    }
  },
  "mapping": {
    "lambotOverridden": [],
    "catchAll": "overflow",
```



```

    "rules": [
      {
        "name": "signature",
        "rule": "$source::$type"
      },
      {
        "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"
      }
    ]
  },
  "ha_profile": "active_active"
},

```

```

    "inputs": [
      {
        "key": "Jane.Doe",
        "value": "MyPassword"
      }
    ],
    "readonly": [
      {
        "name": "url",
        "description": "URL:",
        "value":
"https://example.com/integrations/api/v1/events/webhook2"
      },
      {
        "name": "userid",
        "description": "User ID:",
        "value": "Username"
      },
      {
        "name": "readonly_password",
        "description": "Password:",
        "value": "Password123"
      },
      {
        "name": "auth",
        "description": "Base64 Encoded Auth:",
        "value": "Basic YWRtaW46"
      }
    ]
  ]
}

```

/integrations/{integrationId}/status

The **/integrations/{integrationID}/status** endpoint allows you to check and update the status of an integration.

Back to [Integrations API Endpoint Reference](#).

GET

Retrieves the status of an integration.

Request arguments

The GET request takes the following request argument:

Name	Type	Required	Description
integrationId	Integer	Yes	ID of the integration to retrieve. You can obtain an integration's ID by executing a GET request to /integrations .

Response

The GET request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Description
instances	String	URL of the instance(s) in an Object of Broker IDs that map to the integration status info. For example: <pre>"instances": { "Broker_3a52b7ef_8bad_41cc_8b36_cbb9c2aa3a9e@: { "integration_name": "Azure1", "status": "running", "last_heartbeat": 1567789762645 } }</pre>
global	Object	Contains integration_config_id and status .
integration_config_id	Integer	ID of the integration.
status	String	Current status of the integration.

Examples

The following examples demonstrate making a GET request to the endpoint `/integrations/{integrationId}/status`:

Request example

Example cURL GET request to retrieve the current status of the integration with the ID **"6"**:

```
curl \
https://example.com/integrations/api/v1/integrations/6/status \
-u John.Doe:MyPassword \
-H "accept: application/json"
```

Response example

A successful request returns the HTTP code 200 and no response text.

Example response returning the integration's current status:

```
{
  "global":
  {
    "integration_config_id": 6,"
    "status": "running"
  },
  "instances":
  {
    "Broker_3a52b7ef_8bad_41cc_8b36_cbb9c2aa3a9e":
    {
      "integration_name": "Azure1",
      "status": "running",
      "last_heartbeat": 1567789762645
    }
  }
}
```

PUT

Updates the status of an integration. You can use this to start and stop integrations. An integration can run on multiple brokers; when assigning an integration to a broker, Cisco Crosswork Situation Manager favours the broker(s) running the least integrations.

Request arguments

The PUT request takes the following request argument:

Name	Type	Required	Description
integrationId	Integer	Yes	ID of the integration to update.

Response

The PUT request returns the following response:

Type	Description
HTTP Code	HTTP status or error code indicating request success or failure. See HTTP status code definitions for more information.

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

Name	Type	Required	Description
command	String	Yes	Status command you want to make. Choose from start and stop .
target	String	Yes, unless using the start command	Broker on which to run or stop the integration. If unspecified, Cisco Crosswork Situation Manager starts the integration on the broker running the least integrations.

Examples

The following examples demonstrate making a PUT request to the endpoint **/integrations/{integrationId}/status**:

Request example

Example cURL PUT request to update an integration.

```
curl -X PUT \
https://example.com/integrations/api/v1/integrations/6/status \
-u John.Doe:MyPassword \
{
  "command": "start"
}
```

Response example

A successful request returns the HTTP code 204 and no response text.

Export and Import Integrations

You can use the integrations API to migrate integration configurations across your Cisco Crosswork Situation Manager instances, allowing you to swiftly set up new integrations while keeping previous configurations intact.

This example workflow combines queries to multiple endpoints in order to create an export/import workflow, exporting an integration from one instance and then importing it into another instance.

Before you begin

Before you start the workflow, ensure you have met the following requirements:

1. You have access to two separate instances of Cisco Crosswork Situation Manager and an integration on one of these.
2. You have running brokers on the destination instance.
3. You have the ID of the integration(s) you want to export from your first instance. The ID displays in the URL when you open the integration in your browser. For example, <https://example.com/#/integrations/integration-details/13> has the ID 13.
4. Export the integration

The first step is to export the integration's details. Using basic authentication, make a GET request to your first instance's [/integrations/{integrationId}](#) endpoint to retrieve its payload:

```
curl \
https://instance1.com/integrations/api/v1/integrations/3 \
-u John.Doe:MyPassword \
```

The payload in the response returns the integration's details. For example:

```
{
  "id": 3,
  "name": "DynatraceAPMPolling1",
  "type_id": "dynatrace_apm_lam",
  "version": "2.3",
  "config": {
    "config": {
      "filter": {
        "modules": [],
        "presend": "DynatraceApmLam.js",
        "dependencies": {
          "lambot": [
            "LamBot.js",
            "DynatraceApmLam.js"
          ],
          "contrib": []
        }
      },
      "mapping": {
        "rules": [
          {
            "name": "signature",
            "rule": "$systemprofile :: $rule"
          },
          {
            "name": "source_id",
            "rule": "Dynatrace APM"
          },
          {
            "name": "external_id",
            "rule": "$id"
          }
        ]
      }
    }
  }
}
```

Cisco Systems, Inc. www.cisco.com

```

        {
            "name": "manager",
            "rule": "Dynatrace Apm"
        },
        {
            "name": "source",
            "rule": "$source"
        },
        {
            "name": "class",
            "rule": "$rule"
        },
        {
            "name": "agent",
            "rule": "$LamInstanceName"
        },
        {
            "name": "agent_location",
            "rule": "$LamInstanceName"
        },
        {
            "name": "type",
            "rule": "$state"
        },
        {
            "name": "severity",
            "rule": "$severity",
            "conversion": "sevConverter"
        },
        {
            "name": "description",
            "rule": "$message"
        },
        {
            "name": "agent_time",
            "rule": "$start",
            "conversion": "timeConverter"
        }
    ],
    "catchAll": "overflow",
    "lambotOverridden": [
        "custom_info.overflow",
        "source"
    ]
},
"monitor": {
    "name": "DynatraceApm Lam Monitor",
    "class": "CDynatraceApmMonitor",
    "targets": {
        "target1": {
            "url": "https://localhost:8021",
            "filter": {
                "state": "InProgress",
                "profileName": "nam",
                "incidentRule": "rul"
            }
        },
    },

```

```

        "timeout": 120,
        "password": "def",
        "username": "abc",
        "disable_certificate_validation": true
    },
    },
    "max_retries": -1,
    "retry_interval": 60,
    "request_interval": 60
},
"constants": {
    "severity": {
        "severe": 5,
        "warning": 2,
        "informational": 1
    }
},
"conversions": {
    "stringToInt": {
        "input": "STRING",
        "output": "INTEGER"
    },
    "sevConverter": {
        "input": "STRING",
        "lookup": "severity",
        "output": "INTEGER"
    },
    "timeConverter": {
        "input": "STRING",
        "output": "INTEGER",
        "timeFormat": "yyyy-MM-dd'T'HH:mm:ss.SSS"
    }
},
},
"moolets": [],
"type_id": "dynatrace_apm_lam",
"version": "2.3",
"category": "monitoring",
"ha_profile": "active_passive",
"description": "An integration which enables Moogsoft AIOps to
ingest events from Dynatrace APM.",
"display_name": "Dynatrace APM (Polling)"
},
"inputs": [
    {
        "key": "targets",
        "value": [
            {
                "url": "https://localhost:8021",
                "filter": {
                    "state": "InProgress",
                    "profile_name": "nam",
                    "incident_rule": "rul"
                },
                "password": "def",
                "username": "abc"
            }
        ]
    }
]

```

```

        }
    ],
    {
        "key": "timing",
        "value": {
            "timeout": 120,
            "retry_interval": 60,
            "request_interval": 60
        }
    }
],
"readonly": null
}

```

Import the integration

Now that you have the integration's details from the payload you can import them into your second instance and create a new integration.

Make a POST request to your destination instance's [/integrations](#) endpoint, using the whole payload from the GET request. You do not need to omit the **id** parameter as the POST request ignores it. For example:

```

curl -X POST \
https://instance2.com/integrations/api/v1/integrations \
-u John.Doe:MyPassword \
-d '{
    "id": 3,
    "type_id": "dynatrace_apm_lam",
    "inputs": [
        {
            "key": "targets",
            "value": [
                {
                    "url": "https://localhost:8021",
                    "filter": {
                        "state": "InProgress",
                        "profile_name": "nam",
                        "incident_rule": "rul"
                    },
                    "password": "def",
                    "username": "abc"
                }
            ]
        },
        {
            "key": "timing",
            "value": {
                "timeout": 120,
                "retry_interval": 60,
                "request_interval": 60
            }
        }
    ],
    "name": "DynatraceAPMPolling1",
}

```



```
"version": "2.3"
}'
```

A successful response returns a payload containing the new integration's details.

Start the integration

Having exported the integration, the final step is to start it up.

Make a PUT request to your destination instance's [/integrations/{integrationId}/status](#) endpoint:

```
curl -X PUT \
https://instance2.com/integrations/api/v1/integrations/6/status \
-u John.Doe:MyPassword \
{
  "command": "start"
}
```

The integration is now ready to use on your second instance.

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

See [/document/preview/101040#UIDca48b030717ab676dd20a664b636d681](#) for details of all the MoogDb V2 API methods. MoogDb V2 Method Reference

Methods

All available MoogDb V2 methods are described in the sections below:

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.

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
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",
  "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 occurred

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	<p>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.</p> <p>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.</p> <p>A missing, null, or empty argument implies no sweep-up filter.</p>
first_match_only	Boolean	If True, perform workflow operations once only on each object.
operations	JSON Array	<p>Required</p> <p>A list of operations, each has:</p> <p>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</p> <p>Operation_name - (String, Required for action/decision type) Operation name.</p> <p>function_name - (String, Required for action/decision type) Function name.</p> <p>function_args - (JS Object , action/decision types, only optional) Function arguments.</p> <p>Duration - (Integer, Required for delay type) The number of seconds before the message goes to the next operation/Workflow/moolet.</p> <p>Reset - (Boolean, Required for delay type) Reset the timer on each occurrence?</p>

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

Examples

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. Filter Search Data

Request Argument

Name	Type	Description
filter	String	Filter to match maintenance windows that you want to delete. For example, Description matches " maint_window_12" .
limit	Number	Maximum number of windows to fetch. Defaults 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
```

[deleteTeam](#)

[deleteWorkflow\(\)](#)

Deletes a Workflow mooleet.

Request Arguments

Name	Type	Description
id	Integer	Required -- ID of the workflow to delete.

Response Parameter

Type	Parameter
Boolean	True if the operation succeeded.

findMaintenanceWindows

Find maintenance windows based on a filter and how many should be fetched.

Request Arguments

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
    }
  ]
}
```

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]
}
```

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
    }
  ]
}
```

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

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

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

getSigCorrelationInfo

Retrieves all correlation information related to a specified Situation.

Request Arguments

Name	Type	Description
sitn_id	Number	The Situation ID.

Return Parameters

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 Parameters

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.

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]
  }
}1

```

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: true = get those alerts unique to the Situation 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]
}

```

getSituationFlags

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: true = get those host names unique to the Situation 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"
  ]
}
```

getSituationPrimaryTeam

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"
}
```

getSituationsWithFlag

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	<p>Array of node properties to be returned. Valid properties are:</p> <p>severity: Severity of the node.</p> <p>prc: Whether this node is the probable root cause of the alert.</p> <p>service: Service affected by the node.</p> <p>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.</p> <p>description: Description of the node.</p> <p>vertex_entropy: Vertex Entropy of the node.</p>
fieldName	String	<p>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.</p>

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
}
```

[getToolShares](#)

[getTopPrcDetails](#)

[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	<p>The available functions in the moobot - each key is the function name and the values are:</p> <p>Description (String) -- The description of the function</p> <p>Decision (Boolean) -- If True, treat the result of this function as a decision,.</p> <p>Arguments - (JSON) the arguments of the function, a map from the argument name to:</p> <p>Type - (ENUM/String) the type of argument - either Text, JSON or Number.</p> <p>Description - Human readable description of the argument.</p>
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 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
------	------	-------------

Cisco Systems, Inc. www.cisco.com

situationIds	Native array	A Javascript array containing the IDs of the Situations to merge
keepOriginals	Boolean	Determines what to do with the original Situations: true = keep the original Situations 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](#) for more information.

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 Optional

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

rateSituation

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

reload

Takes a situation (CMooBotSituation) or alert (CMooBotAlert) type of CEvent and refreshes the data in the CEvent payload but preserves the metadata. This method should be used instead of getSituation and getAlert if you want to update the event with the latest data from the database, and when you are forwarding an event on using **situation.forward(<event>)**.

Request Arguments

Name	Type	Description
event	CEvent	A CEvent object representing the Alert, containing Alert attributes, such as type, severity, etc.

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

removeSituationPrimaryTeam

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.

		Note: 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. Note: 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: true = merge the existing data with the new data 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: 0 Clear 1 Indeterminate 2 Warning 3 Minor 4 Major 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 non_causal 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: true = merge the existing data with the new data false = replace the existing data with the new data

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

[setSituationFlags](#)

[setSituationPrimaryTeam](#)

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

[shareToolAccess](#)

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: 1 for a recurring maintenance window. 0 for a one-time maintenance window. 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: 0 = a one-time maintenance window 2 = daily 3 = weekly 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,
```

Cisco Systems, Inc. www.cisco.com

```

    "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: <pre>[{ "name" : "SunOS" , "ranking" : 40 }, { "name" : "SAP" , "ranking" : 50 }, { "name" : "EMC" , "ranking" : 60 }]</pre>
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	<p>A list of operations, each being:</p> <p>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</p> <p>operation_name - (String, Required for action/decision type) The operation name.</p> <p>* function_name - (String, Required for action/decision type) The function name.</p> <p>* function_args - (JS Object) (<i>Optional for action/decision type</i>) The arguments for the action or decision function.</p> <p>duration - (Integer) (Required for delay type) The amount of seconds before the message goes to the next operation/Workflow/moolet</p>
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"});
```

checkSituationFlag

A MoogDb v2 method that checks whether a flag is associated with a Situation.

See [Situation Flags](#) for more information on Cisco Crosswork Situation Manager Situation flags.

Request arguments

Method **checkSituationFlag** takes the following request arguments:

Name	Type	Required	Description
sitn_id	Number	Yes	ID of the Situation to be checked.

flag	String	Yes	Flag to be checked for the specified Situation ID.
-------------	--------	-----	--

Response

Method **checkSituationFlag** returns the following response:

Type	Description
Boolean	Indicates whether or not the operation was successful: true = success.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **checkSituationFlag**:

Request example

Example request to check whether Situation 1 contains the flag S1:

```
var result = JSON.stringify(moogdb.checkSituationFlag(1, "S1"))
```

Response example

A successful request returns **true**.

Example response returning **true** because the Situation contains the specified flags:

```
true
```

deleteTeam

A MoogDb v2 method that deletes a single team.

Request arguments

Method **deleteTeam** takes the following request arguments:

Name	Type	Required	Description
team_id	Number	Yes	ID of the team you want to delete.

Response

Method **deleteTeam** returns the following response:

Type	Description
------	-------------

Boolean	Indicates whether or not the operation was successful: true = success.
---------	---

Examples

The following examples demonstrate typical use of method **deleteTeam**:

Request example

Example request to delete a team with ID 33.

```
var success = moogdb.deleteTeam(33)
```

Response example

A successful request returns **true**.

getProcesses

A MoogDb v2 method that returns a list of processes from the database.

Request arguments

Method **getProcesses** takes the following request arguments:

Name	Type	Required	Description
limit	Integer	No	Maximum number of processes to return. Default is 1000.
query	String	Yes	A JSON or SQL like filter of the process name.
exact_match	Boolean	No	If true , the query performs an exact match on the process name. If false , the query checks for contains only on the process name. Default is false .

Response

Method **getProcesses** returns the following response:

Type	Description
Object	A list of strings describing the requested processes, or a null value if there is an error.

Examples

The following examples demonstrate typical use of method **getProcesses**:

Request example

Example request to return the first thousand process names containing "Network":

```
var actions = moogdb.getProcesses(1000, "Network", false);
```

Response example

Example response returning returning details of all process names containing "Network":

```
[
  {
    "process_id": 1,
    "name": "Network LON",
    "description": "Network London"
```

```

    },
    {
      "process_id": 2,
      "name": "NY Network A",
      "description": "Network New York A"
    },
    {
      "process_id": 3,
      "name": "NY Network B",
      "description": "Network New York B"
    }
  ]

```

getServices

A MoogDb v2 method that returns a list of services from the database.

Request arguments

Method **getServices** takes the following request arguments:

Name	Type	Required	Description
limit	Integer	No	Maximum number of services to return. Defaults to 1,000.
start	Integer	No	Number of the first service to return. Defaults to 0.
query	String	Yes	A JSON or SQL like filter of the service name.
exact_match	Boolean	No	If true , the query performs an exact match on the service name. If false , the query checks for contains only on the service name. Defaults to false .

Response

Method **getServices** returns the following response:

Type	Description
Native Object	A list of strings describing the requested services, or a null value if there is an error.

Examples

The following examples demonstrate typical use of method **getServices**:

Example Using Exact Matching

Example request using exact matching of the query "Network LON":

```
var actions = moogdb.getServices(0, 1000, "Network LON", true);
```

Example response returning details of the service name "Network LON":

```

[ {
  "service_id": 3,
  "name": "Network LON",
  "description": "Network description"
}]

```

Example Using Approximate Matching

Example request using approximate matching of the query "Network":

```
var actions = moogdb.getServices(0, 1000, "Network", false);
```

Example response returning details of all service names containing "Network":

```
[{
  "service_id":1,
  "name":"Network LON",
  "description":"Network London"
},{
  "service_id":2,
  "name":"NY Network A",
  "description":"Network New York A"
},{
  "service_id":3,
  "name":"NY Network B",
  "description":"Network New York B"
}]
```

getSituationFlags

A MoogDb v2 method that returns the flags for one or more Situations.

See [Situation Flags](#) for more information on Cisco Crosswork Situation Manager Situation flags.

Request arguments

Method **getSituationFlags** takes the following request arguments:

Name	Type	Required	Description
sitn_ids	Array	Yes	IDs of the Situations to return the flags for.

Response

Method **getSituationFlags** returns the following response:

Type	Description
JSON Array	An array of the flags associated with the specified Situation IDs.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **getSituationFlags**:

Request example

Example request to return the flags associated with Situation IDs 1 and 2:

```
var result = JSON.stringify(moogdb.getSituationFlags([1, 2]))
```

Response example

Example response returning the flags associated with specified Situations:

```
{"1":["A1", "B1"], "2":["A1"]}
```

getSituationPrimaryTeam

A MoogDb v2 method that returns the primary team on the specified Situation.

Request arguments

Method **getSituationPrimaryTeam** takes no request arguments.

Name	Type	Required	Description
sitn_id	Number	Yes	ID of the Situation you want to retrieve the primary team for.

Response

Method **getSituationPrimaryTeam** returns the following response:

Type	Description
Object	A Javascript object containing the Situation ID and the primary team ID.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **getSituationPrimaryTeam**:

Request example

Example request to return the primary team for Situation 1906:

```
var actions = moogdb.getSituationPrimaryTeam(1906);
```


Response examples

Example response returning that team 36 is the primary team for Situation 1906:

```
{
  "primary_team_name": "Infrastructure",
  "sitn_id":1906,
  "primary_team_id":36
}
```

Example response returning that Situation 1906 does not have a primary team assigned to it:

```
{
  "sitn_id":1906,
}
```

getSituationsWithFlag

A MoogDb v2 method that returns all the Situations which have the specified flag.

See [Situation Flags](#) for more information on Cisco Crosswork Situation Manager Situation flags.

Request arguments

Method **getSituationsWithFlag** takes the following request arguments:

Name	Type	Required	Description
flag	String	Yes	Flag to search for.
start	Number	No	Starting point of the result set to return. Default is 0.
limit	Number	No	Number of results to return. Default is 1000.

Response

Method **getSituationsWithFlag** returns the following response:

Type	Description
JSON Array	An array of the Situation IDs that have the specified flag associated with them.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **getSituationsWithFlag**:

Request example

Example request to return all Situations with flag "S1":

```
var result = JSON.stringify(moogdb.getSituationsWithFlag("S1",0,1000))
```

Response example

Example response returning the Situation IDs with the specified flag:

```
[1, 2, 3]
```

getToolShares

A MoogDb v2 method that returns the shared access for a specified tool.

Request arguments

Method **getToolShares** takes the following request arguments:

Name	Type	Required	Description
tool_id	Number	Yes	ID of the tool that you want to retrieve its shared access for.

Response

Method **getToolShares** returns the following response:

Type	Description
Object	A Javascript object containing the tool ID, the domain, and an array of all the domain IDs that can access the tool.

Examples

The following examples demonstrate typical use of method **getToolShares**:

Request example

Example request to retrieve all the domain IDs that have access to tool 15 :

```
var actions = moogdb.getToolShares(15);
```

Response example

Example response returning that tool ID 15 can be accessed by team ID 3:

```
{
  "tool_id": 15,
  "domain_ids": [
    3
  ],
  "domain": "team"
}
```

getTopPrcDetails

A MoogDb v2 method that returns the top most likely causal alerts, based on their Probable Root Cause value, for a specified Situation.

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, for the specified Situation, irrespective of whether they have been labeled causal or are unlabeled. Alerts marked as symptoms are excluded from the return.

Back to </document/preview/101040#UIDca48b030717ab676dd20a664b636d681>.MoogDb V2
Method Reference

Request arguments

Method **getTopPrcDetails** takes the following request arguments:

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 unlabeled alerts to return. Default is 1, if not specified, returning one alert with the highest root cause probability.

Response

Method **getTopPrcDetails** returns the following response:

Type	Description
JSON Array	An array of objects containing the details of the causal or unlabeled alerts with the highest root cause probability in the specified Situation.

The following details are returned for each alert:

Name	Type	Description
rc_probability	Number	Root cause probability of the alert.
description	String	Description of the alert.
rc_label	Integer	Label defining whether the alert is causal or unlabeled. Alerts marked as symptoms are excluded from the return. 1 = causal 0 = unlabeled -1 = symptom
alert_id	Integer	Alert ID.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

This method does not return anything if the alerts have been archived to the historic database even if the Situation is still in the active database.

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **getTopPrcDetails**:

Request example

Example request to return the top three causal alerts with the highest root cause probability in Situation 145:

```
var result = JSON.stringify(moogdb.getTopPrcDetails(145,3))
```

Response example

Example response returning the top three causal or unlabeled alerts for Situation ID 145:

```
{
  "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
    }
  ]
}
```

rateSituation

A MoogDb v2 method that applies a rating to a specified Situation.

Request arguments

Method **rateSituation** takes the following request arguments.

Name	Type	Required	Description
situationId	Number	Yes	ID of the Situation you want to rate.
rating	Number	Yes	Rating that you want to apply to the Situation. This is equivalent to the number of stars that you can assign to a Situation in the UI. One of: 0 = Not yet rated 1 = Bad 2 = Poor 3 = Adequate 4 = Good 5 = Excellent
comment	String	No	A comment about the rating you are applying to the Situation.

Response

Method **rateSituation** returns the following response:

Type	Description
Object	A Javascript object containing rating , comment , and sig_id of the rated Situation.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **rateSituation**:

Request example

Example request to apply a rating of 4 to Situation ID 18 with a comment "Rating 4":

```
var success = moogdb.rateSituation(18, 4, "Rating 4");
```

Response example

Example response returning the rating number, comment and ID of the rated Situation:

```
{"rating":4,"comment":"Rating 4","sig_id":18}
```

removeSituationPrimaryTeam

A MoogDb v2 method that removes the primary team from a Situation. The team remains assigned to the Situation.

Request arguments

Method **removeSituationPrimaryTeam** takes the following request arguments:

Name	Type	Required	Description
sitn_id	Number	Yes	ID of the Situation that you want to remove the primary team from.

Response

Method **removeSituationPrimaryTeam** returns the following response:

Type	Description
Object	A Javascript object containing the Situation ID.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **removeSituationPrimaryTeam**:

Request example

Example request to remove the primary team from Situation 1906:

```
var actions = moogdb.removeSituationPrimaryTeam(1906);
```

Response example

Example response returning the Situation ID that the primary team has been removed from:

```
{
  "sitn_id": 1906
}
```

setSituationFlags

A MoogDb v2 method that updates the flags associated with a specified Situation. You can add flags to or remove them from a Situation.

See [Situation Flags](#) for more information on Cisco Crosswork Situation Manager Situation flags.

Request arguments

Method **setSituationFlags** takes the following request arguments:

Name	Type	Required	Description
sitn_ids	Array of Numbers	Yes	An array of IDs for the Situations you want to update.
to_add	Array of Strings	Yes	Flags to be added to those Situations. If this is an empty list, no flags are added to the Situation.
to_remove	Array of Strings	Yes	Flags you want to remove from the Situation. If this is an empty list, no flags are removed from the Situation.

Response

Method **setSituationFlags** returns the following response:

Type	Description
Boolean	Indicates whether or not the operation was successful: true = success.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **setSituationFlags**:

Request example

Example request to update Situation IDs 1 and 2:

```
var result = JSON.stringify(moogdb.setSituationFlags([1, 2], ["S1","S2"], ["S2"]))
```

Response example

A successful request returns **true**.

setSituationPrimaryTeam

A MoogDb v2 method that sets one of the teams already assigned to a Situation as the primary team.

Request arguments

Method **setSituationPrimaryTeam** takes the following request arguments:

Name	Type	Required	Description
sitn_id	Number	Yes	ID of the Situation.
team_id	Number	No, if team_name is used.	ID of the team that you want to make the primary team.
team_name	String	No, if team_id is used.	Name of the team that you want to make the primary team.

Response

Method **setSituationPrimaryTeam** returns the following response:

Type	Description
Boolean	Indicates whether or not the operation was successful: true = success.
Type	Description
Object	A Javascript object containing the Situation ID and the primary team ID.

API update behavior

The behavior of this method depends on whether the relevant Situation is open, closed and still in the active database, or closed and archived to the historic database. This method updates the Situation as follows:

Situation Status	API Updates Situation
Open Situation	Yes
Closed Situation in active database	Yes
Closed Situation in historic database	No

See [API Update Behavior](#) for more information on Situation statuses.

Examples

The following examples demonstrate typical use of method **setSituationPrimaryTeam**:

Request example

Example request to set the team "Database Management System" as the primary team on Situation 1906:

```
var actions = moogdb.setSituationPrimaryTeam(1906, 12);
```

Response example

Example response returning that team 12 is the primary team on Situation 1906:

```
{
  "primary_team_name": "Infrastructure",
  "sitn_id": 1906,
  "primary_team_id": 12
}
```

shareToolAccess

A MoogDb v2 method that shares access to a tool with other users, teams, or roles, or makes it global so that all users can access it. When a user creates a tool, it is automatically shared globally. You can use this endpoint to restrict its availability and ensure that tools are only available to users who need them. Using this endpoint to share access to a tool overwrites any existing shares.

Request arguments

Method **shareToolAccess** takes the following request arguments:

Name	Type	Required	Description
tool_id	Number	Yes	ID of the tool that you want to share access for.
domain	String	Yes	Domain to share access with. One of: user , team , role , or global .
domain_ids	Array	Yes/No	An array of one or more IDs within the domain. Optional for the global domain.

Response

Method **shareToolAccess** returns the following response:

Type	Description
Boolean	Indicates whether or not the operation was successful: true = success.

Examples

The following examples demonstrate typical use of method **shareToolAccess**:

Request example

Example request to share access of tool ID 15 with team ID 3:

```
var actions = moogdb.shareToolAccess(15, "team", 3);
```

Response example

A successful request returns **true**.

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. So 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.
Logger Configure 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 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:

1. Change and create structure in the moogdb database
2. Listen for specific Events in the system
3. Push Events out
4. Log to the common log file output
5. 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:

1. Add the external JavaScript module file (**BotExampleModule.js**) in the **\$MOOGSOFT_HOME/bots/moobots** or the **\$MOOGSOFT_HOME/contrib** directory
2. 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:
'0rJG15oCWpmE9Hbk32sxFgxlQV3O5cx2bx1vKNOM7YA='
  }
};
}
```

Config

1. [Before You Begin](#)
2. [Best Practice](#)
3. [Error Reporting](#)
4. [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:

1. The configuration file is in valid JSON
2. The configuration file is in **\$MOOGSOFT_HOME/config**
3. 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:

1. Use the module within the constraints of the **OnLoad** function.
2. Note that making multiple calls to the module may impact the performance of the bot.
3. Keep custom configuration files in a subdirectory of **\$MOOGSOFT_HOME/config** and name them appropriately.
4. 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

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**.

1. **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);
```
2. **get** takes the name of the shared attribute **counter**.
3. **count_val** is incremented.
4. **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');
```

Note: Compatibility with MoogDb and MoogDb V2 Methods and auxiliary objects listed here are compatible with the [MoogDb V2](#) module.

Method

1. 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() method

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();
```

1. Call the [newEvent](#) JavaScript function.
2. Define the Event type Event (from the [Constants](#) module), which responds to events put on the Message Bus by a LAM.
3. Call the [listen](#) function in-line to listen for the event type.

When the Moolet starts and loads this events Moobot, its 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.

Event forward methods

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).

alert.forward("Cookbook");

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):

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**.

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 .

CEvents API

Cisco Crosswork Situation Manager uses the CEvents API to pass data to LAMbot functions.

This API uses the following methods.

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. Returns true if the filter matches the event, alert or Situation. Returns false if the filter has a correct syntax but doesn't match the event, alert or Situation. Returns null if the filter syntax is incorrect.

Example

```
var is_matching = situation.evaluateFilter("description LIKE 'Created Situation'");
```

forward(*moobot*)

Forwards 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*,...)

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	<p>The summary of information about a system:</p> <p>summary.alert_count - number</p> <p>summary.service_count - number</p> <p>summary.sig_summaries - map (contains "categories" and "queues")</p> <p>summary.sig_summaries.categories - (array of objects)</p> <p>summary.sig_summaries.queues - (array of objects)</p> <p>categories and queues contain the following:</p> <p>sig_total - number, alert_total - number, name - string</p> <p>summary.sigs_down - number</p> <p>summary.sigs_up - number</p> <p>summary.total_events - number</p> <p>summary.total_sigs - number</p>

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.

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	The whole custom info object to 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.

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:

1. custom_info
2. description
3. first_event_time
4. last_event_time
5. 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();
```

1. Call the `newEvent` JavaScript function.
2. Define the event type **event** (from the `Constants` module), which responds to events put on the Message Bus by a LAM.
3. Call the `listen` function 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()

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:

1. Bot.isActive: Returns whether the specified Moolet is active or not.
2. 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

1. [Description](#)

a) [Methods](#)

2. [Reference Guide](#)

a) [externalDb.connect\(\)](#)

b) [externalDb.execute\(\)](#)

c) [externalDb.query\(\)](#)

d) [externalDb.prepare\(\)](#)

3. [Database Specific Information](#)

a) [Downloading JDBC Drivers](#)

b) [Microsoft SQL Server](#)

c) [MySQL](#)

d) [IBM DB2](#)

e) [Oracle](#)

f) [PostgreSql](#)

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):

- 1) MySQL
- 2) Microsoft SQL Server
- 3) IBM DB2
- 4) Oracle
- 5) 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.

Methods

- 1) [externalDb.connect](#)
- 2) [externalDb.execute](#)
- 3) [externalDb.query](#)
- 4) [externalDb.prepare](#)

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. 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: MySQL - 3306 SQL Server - 1433 DB2 - 50000 Oracle - 1521 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. Defaults: SQL Server - sqljdbc4.jar DB2 - db2jcc4.jar Oracle - ojdbc6.jar PostgreSql - postgresql-9.3-1102.jdbc41.jar Assumes it will find these files in \$MOOGSOFT_HOME/lib/cots/ They are not bundled in a regular Moogsoft AIOps installation.
class_name	The name of the JDBC class. Defaults: SQL Server - com.microsoft.sqlserver.jdbc.SQLServerDriver MySQL - com.mysql.jdbc.Driver DB2 - com.ibm.db2.jcc.DB2Driver Oracle - oracle.jdbc.OracleDriver 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

	<p>the external database by including the pool_size key.</p> <p>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.</p>
--	--

Note: You can also define connection properties in the following configuration file:

moog_external_db_details.conf

Return Parameter

Type	Description
Object	<p>A Java object containing connection details, depending on the requested connection properties</p> <p>Returns null if no connection is available (due to either misconfiguration or unavailability of the external database).</p>

Notes: 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: 'wrPass'});
```

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 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.

	Note: 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: Be sure to download the correct version of the JDBC Driver for your database. 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](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>

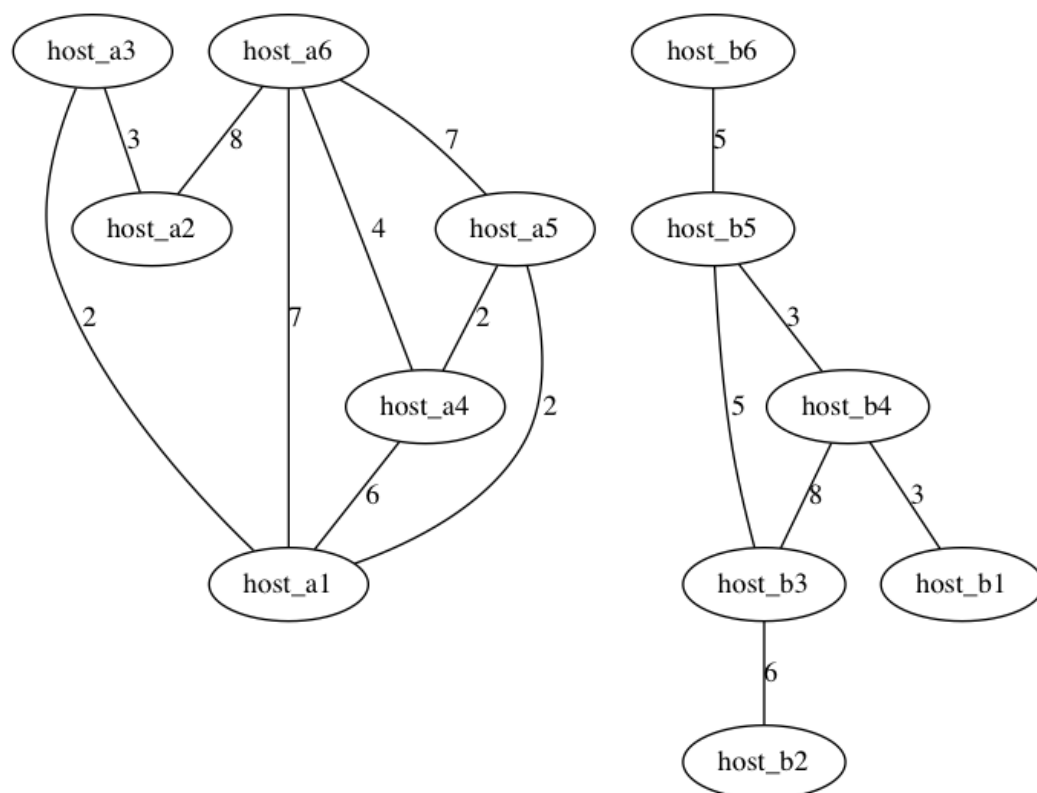
If no **<weighted edge number>** is included, the default value of 1 is used.

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. The maximum Distance to return a result for

Return Parameter

Type	Description
Number	The Distance between the two nodes. Returns -1 if: a node is not in the topology or the two nodes are not directly or indirectly connected 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)

Cisco Systems, Inc. www.cisco.com

These will add a new node to a topology/graph both in memory and in the database.

Behavior:

1. if unspecified, weight will have default value 1.0
2. any new nodes will saved in memory and db
3. new connection will be saved in memory and db

GraphTopo:

1. won't work if there already is such edge
2. uses jgraph methods addVertex and addEdge

Topo:

1. won't work if both nodes aren't in topology or if both nodes already are in
2. does not recalculate a topology, new coordinate == old coordinate + weight
3. new coordinates will be saved in memory and database

Logger

1. `logger.debug()`
2. `logger.info()`
3. `logger.warning()`
4. `logger.fatal()`

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.

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

- 1) `mailer.initTransport(mailerObj)`
- 2) `mailer.send(mailMsg)`

`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: Please 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);
```

Moolet Informs

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:
 - a) `inform.setSubject`: Subject of the inform message. You can use this to enable a different workflow within the target Moobot.
 - b) `inform.setPayload`: Any CEvent object. See [Events](#) for more information.
 - c) `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:
 - a) `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 as sending data.

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: Bot.ALERT_BUILDER

	Bot.ALERT_RULE_ENGINE
	Bot.COOKBOOK
	Bot.EMPTY_MOOLET
	Bot.NOTIFIER
	Bot.SCHEDULER
	Bot.SITUATION_MANAGER
	Bot.TEAMS_MANAGER
	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() === 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: Bot.WorkflowEngine.ALERT Bot.WorkflowEngine.SITUATION Bot.WorkflowEngine.EVENT 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. It is available to load into any standard Moobot.

To use it, define a new global object at the top of a Moobot JavaScript file to load the module. For example:

```
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 Moogfarmd, 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 Moogfarmd.

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

1. [Configure the Module](#)

2. [Reference Guide](#)

- a) [connect](#)
- b) [send](#)
- c) [close](#)

3. [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:

- Define a new global object **rabbit** at the top of a Moobot JavaScript file to load the module.
- Use the **connect** method to create a new connection to one or more RabbitMQ brokers.
- Use the **send** method to send the required information.
- 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: host : Hostname or IP address of the RabbitMQ broker. port : Port of the RabbitMQ broker.
user	Username to connect to RabbitMQ.
password	Password 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. Refer to Message System SSL for more information.

Return Parameter

Type	Description
Object	A Java object containing connection details, depending on the requested connection properties. 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 `basicclass` 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: Plain text JSON Object payload 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	The request is sent asynchronously, returns null and the callback

	ck functio n	function is called regardless of the success or failure of the request. See below.
success	Callba ck functio n	The request is sent asynchronously, returns null and the success function is called only the request was successful. See below.
failure	Callba ck functio n	The request is sent asynchronously, returns null and the failure function is called only if the request failed. See below.
timeout	Numb er	The period of time (in seconds) to wait for response before completing with timeout error. If 0 or less, wait indefinitely. The default is 120 seconds.
proxy	String or Object	Host, port, user, encrypted_password/password. For example, as an object: <pre>proxy:{ host:"proxyhost", port:1223, user:"proxyuser", encrypted_password:"2KctaEbJH/m8rz4WqgmZYZfdripdIsku 7fOFJWM6YNA=" //password: "unencrypted_plain_text_password" }</pre> As an object, you can either specify a Cisco encrypted password or a plain text password, specifying both will favour the encrypted_password value. Or, as a string, where format is <user>:<password>@<host>:<port> <pre>proxy: "proxyuser:passw0rd@proxyhost:1223"</pre> 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 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.

	n	
failure	Callba ck functio n	The request is sent asynchronously, returns null and the failure function is called only if the request failed. See below.
timeout	Numb er	The period of time (in seconds) to wait for response before completing with timeout error. If 0 or less, wait indefinitely. The default is 120 seconds.
proxy	String or Object	Host, port, user, encrypted_password/password. For example, as an object: <pre> proxy:{ host:"proxyhost", port:1223, user:"proxyuser", encrypted_password:"2KctaEbJH/m8rz4WqgmZYZfdripdIsku7fOFJWM6YNA=" //password: "unencrypted_plain_text_password" } </pre> As an object, you can either specify a Cisco encrypted password or a plain text password, specifying both will favour the encrypted_password value. Or, as a string, where format is <user>:<password>@<host>:<port> proxy : "proxyuser:passw0rd@proxyhost:1223" 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. (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	The request is sent asynchronously, returns null and the success function is called only the request was successful. See

	function	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. 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. (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.

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. 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. (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.

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","headers":{"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://surveillanceserver_84:9090/reportAntiSoc",
        params: {
            crime: "Graffiti"
        },
        proxy: {
            host: "proxyserver",
            port : 3128,
            user : "username",

encrypted_password: "zm01xjTGIAhp6LrpM49+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:

```

a. 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:

```

a. 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');
```

LAMbots

```
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:

1. You have the details of the API to query.
2. 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**.

1. Configure the behavior of the LAM:
 - a) `request_interval`: Length of time to wait between calls to the `execute` method, in seconds. Defaults to 60.
 - b) `num_threads`: Number of worker threads to use for processing events. Defaults to 5.
2. Optionally configure the LAM identification and logging details in the `agent` and `log_config` sections of the file:
 - a) `name`: Identifies events the LAM sends to the Message Bus.
 - b) `capture_log`: Name and location of the LAM's log file.
 - c) `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.