



Cisco Crosswork Situation Manager 7.0.x Configuration Guide

(Powered by Moogsoft AIOps 7.0.1)

1. OpsConfiguration.....	IOps2
1.1 Cisco Crosswork Situation Manager Component Logs	2
1.2 Archive Situations and Alerts	4
1.2.1 Archiver Command Reference	6
1.3 Authentication	7
1.3.1 SAML 2.0	7
1.3.2 Security Configuration.....	8
1.3.3 Configure Single Sign-On with SAML.....	22
1.3.3.1 Build a Service Provider Metadata File	27
1.3.3.1.1 Service Provider Metadata Reference.....	28
1.3.3.2 Security Configuration Reference.....	29
1.4 Configure the Tool Runner	31
1.4.1 Tool Runner	38
1.5 Custom Info.....	42
1.6 Data Archiving.....	45
1.7 Data Parsing	45
1.8 Determine Probable Root Cause.....	49
1.8.1 Probable Root Cause Configure and Retrain	50
1.9 Event and Alert Field best practice	52
1.10 Event Processing	58
1.10.1 MooBot Modules	60
1.10.1.1 Config Bot Module.....	63
1.10.1.2 Constants.....	64
1.10.1.3 Events.....	67
1.10.1.3.1 CEvents API	70
1.10.1.3.2 Events (MOOGDb only)	76
1.10.1.4 ExternalDb	82
1.10.1.5 Graph Topology	90
1.10.1.6 Logger	95
1.10.1.7 Mailer	98
1.10.1.8 MoogDb V2	100
1.10.1.9 Process.....	139
1.10.1.10 REST.V2	141
1.10.1.11 Utilities	155
1.10.2 Moollets.....	158
1.10.2.1 Alert Builder Moollet	159
1.10.2.2 Alert Rules Engine.....	160
1.10.2.2.1 Heartbeat Monitor	163
1.10.2.2.2 Status ID Reference.....	167
1.10.2.3 Classic Sigaliser.....	167
1.10.2.4 Empty Moollet	171
1.10.2.4.1 Invoking custom functionality	172
1.10.2.4.2 Situation Actions	173
1.10.2.5 Enricher Moollet	173
1.10.2.6 Housekeeper Moollet	175
1.10.2.7 Maintenance Manager Moollet	176
1.10.2.8 Notifier Moollet	179
1.10.2.9 Scheduler Moollet	179
1.10.2.10 Situation Manager	181
1.10.2.11 Speedbird Moollet	183
1.10.2.12 Template Matcher Moollet	189
1.10.2.13 Teams Manager Moollet.....	190
1.11 Graze API.....	191
1.11.1 Situation Action Codes.....	277
1.11.2 Stats API.....	277
1.12 Historic Database	336
1.12.1 Historic Database Benefits	339
1.12.2 Historic Data Utility Command Reference	341
1.13 Import a Network Topology.....	342
1.13.1 Topology Builder Command Reference	343
1.14 Logging	343
1.15 Message Bus	343
1.15.1 Message System Deployment.....	345
1.15.2 Message System Troubleshooting	347
1.15.3 Message System SSL.....	351
1.16 Moog Encryptor	354
1.17 Search and Indexing.....	356
1.18 Situation Merge Behavior	358
1.19 Situation Room Plugins	360
1.20 SMS Configuration	362
1.21 Troubleshooting.....	365
1.22 URL-based Filters	379
1.23 Valid SSL Certificates.....	386

Configuration

The Configuration section has information about configuration including the Graze API, plugins, archiving and tools.

Click the links below for more information:

Cisco Crosswork Situation Manager Component Logs

Cisco Crosswork Situation Manager components generate log files to report their activity. As a Cisco Crosswork Situation Manager administrator, you can refer to the logs to audit system usage or diagnose issues. In certain cases you may want to change logging levels based upon your specific environment or needs.

See [Logger](#) for information on the Logger Moobot module.

Configure Your Log Files

By default moogfarmd writes logs into a log file stored in \$MOOGSOFT_HOME/var/log/moogsoftif you have write permissions for this directory. Otherwise, the logs are written to \$MOOGSOFT_HOME/log. The log file takes the name of the HA address of the process. For example, MOO.moog_farmd.farmd_instance1.log.

To save the logs to a different file, run moogfarmd with this option:

```
service moogfarmd start --logfilename <filename>
```

To run the logs on the console only, run moogfarmd with this option:

```
service moogfarmd start --logconsole
```

Log Files by Cisco Crosswork Situation Manager Component

The following reference provides the information about the log files for the various Cisco Crosswork Situation Manager components.

Apache Tomcat

Log location: /usr/share/apache-tomcat/logs

Primary log file: catalina.out

To change the logging level for the Cisco Crosswork Situation Manager servlets which run in Tomcat:

1. Edit \$MOOGSOFT_HOME/config/servlets.conf.
2. Set the default log level using the top-level loglevelproperty.

The available logging levels are ALL, INFO, WARN, DEBUG, and NONE. For example, to enable all logging:

```
loglevel: "ALL",
```

3. You can optionally override the default log level using the loglevelproperty specific to the following objects: moogsvr, moogpoller, toolrunner, graze, and events. The available log levels are the same as the default.
4. Restart Apache Tomcat.

Nginx

Log location: /var/log/nginx

Primary log file: error_log

To change the logging level for Nginx:

1. Edit /etc/nginx/conf/nginx.conf.
2. Set the LogLevel property. For example to enable debug logging:

```
LogLevel debug
```

3. Restart Ngnix.

Integrations (LAMs)

Default log location: \$MOOGSOFT_HOME/log/data-capture

For example \$MOOGSOFT_HOME/log/data-capture/rest_client_lam.log

Some LAMs specify other log locations.

moogfarmd

Log location: /var/log/moogsoft/moogfarmd.log

To change the logging level for moogfarmd:

1. Edit /etc/init.d/moogfarmd.
2. Set the LogLevel property. The available logging levels are ALL, INFO, WARN, DEBUG, and NONE. For example to enable debug logging:

```
LOG_LEVEL=DEBUG
```

3. Restart moogfarmd.

MySQL

Log location: /var/log/mysqld.log

MySQL logging is defaults to the highest level. To remove warning from the MySQL log:

- 1 Edit /etc/my.cnf.
- 2 Add the following line:

```
log_warnings = 0
```

- 3 Restart the mysqld service.

RabbitMQ

Log location: /var/log/rabbitmq

Refer to the [RabbitMQ documentation](#) for information on how to configure RabbitMQ.

Elasticsearch

Log location: /var/log/elasticsearch/elasticsearch.log

Refer to the [Elasticsearch documentation](#) for information on how to configure Elasticsearch.

Log Rotation for Cisco Crosswork Situation Manager Components

Moogfarmd and integrations (LAMs) use a Java-based logging utility that automatically runs at startup to prevent log files becoming unmanageably large. The utility also prevents the loss of log data when you restart Cisco Crosswork Situation Manager.

The utility compresses each rotated log into gzip (.gz) format and appends the filename with a date stamp. Rotated log files are retained for 40 days before they are purged.

The logging utility rotates the logs when the file size reaches 500MB by default. It rotates up to 40 files by default. This is controlled in by two parameters in \$MOOGSOFT_HOME/config/system.conf:

file_size

The size limit of the log file in megabytes that triggers a log rotation.

Type: Integer

Default: 500M

maximum_files

The maximum number of files that Cisco Crosswork Situation Manager can rotate.

Type: Integer

Default: 40

The default logger configuration appears in \$MOOGSOFT_HOME/config/system.conf as follows:

```
# Logger configuration # ,  
# "logger":  
# {  
#     "file_size" : "500M", #  
#     "maximum_files" : "40" # }
```

Archive Situations and Alerts

You can run the command-line archiver tool included with Cisco Crosswork Situation Manager to archive and delete Situations, alerts, and statistical data. The benefits of archiving data include improved system performance, faster backup and recovery, reduced maintenance, and lower storage costs.

How Archiving Works

The archiver tool archives and deletes a single day's worth of data at a time, to reduce the impact on the database. After you launch the archiver, it automatically processes data in batches which are configurable using the -b, -y and -z options in the [Archiver Command Reference](#).

Both the moogsoft-db and moogsoft-utils packages include the archiver tool. You can find it at:

\$MOOGSOFT_HOME/bin/utils/moog_archiver

The archiver exports and deletes data from the historic database, unless you are deleting statistical data which resides only in the active database. If the [historic database](#) is disabled it performs all operations against the active database.

By default the archiver writes files to the /usr/local/archiveddirectory.

Launch the Archiver

To launch the archiver execute the moog_archiver command and pass either the -e argument to export or the -r option to delete.

Export all data older than 28 days to the default directory and retain the data in the database:

```
./moog_archiver -e
```

Delete all data older than 28 days:

```
./moog_archiver -r
```

See the [Archiver Command Reference](#) for a full list of available arguments.

Archive Loose Alerts

You can modify the selection criteria for loose alerts and Situations and their member alerts. You can choose to archive and delete loose alerts only using the last example below.

Export loose alerts that have not been modified in the past 28 days, and closed/dormant/superseded Situations and their member alerts that have not been modified in the past 4 days, and then delete the data from the database:

```
./moog_archiver -e -r -o -s 4
```

Export loose alerts that have not been modified in the past 2 days, and closed/dormant/superseded Situations and their member alerts that have not been modified in the past 7 days, and then delete the data from the database:

```
./moog_archiver -e -r -o -l 2 -s 7
```

Export loose alerts that have not been modified in the past 28 days, and then delete the data from the database:

```
./moog_archiver -e -r -t
```

Archive Filtered Situations and Alerts

You can use global Situation and alert filters to limit the data that is eligible for archiving and deletion.

Export loose alerts that have not been modified in the past 28 days, and Situations and their member alerts that have not been modified in the past 7 days and match the global filter "My Global Alert Filter", and then delete the data from the database:

```
./moog_archiver -e -r -s 7 -i "My Global Alert Filter"
```

Delete all Situations that match the filter "My Global Situation Filter" and their member alerts, and delete all loose alerts that match the filter "My Global Alert Filter":

```
./moog_archiver -r -s 0 -l 0 -i "My Global Situation Filter" -a "My Global Alert Filter"
```

Use filters that extract data based on age with caution, as they can conflict with specified (or default) age constraints. If you use a filter that selects Situations created during the past day and apply an option to archive Situations older than 28 days, no data will be archived.

Delete Situations, Alerts and Statistical Data

You can use the archiver to delete Situations, alerts and statistical data that match specified criteria from the database.

Delete all Situation and alert data:

```
./moog_archiver -r -s 0 -l 0
```

Delete statistical data older than 15 days:

```
./moog_archiver -m -n 15
```

Delete files older than 7 days from the default directory:

```
./moog_archiver -f 7
```

Archive File Names and Structure

Archive files are named and structured as follows:

- Archive files containing Situation data including alerts, events and snapshots have the filename format <table name>-<yyyymmdd>. <hhmmss>.csv.
For example alerts-20150410.143637.csv
- Archive files containing loose alert data have the filename format <table name>-loose<yyyymmdd>.<hhmmss>.csv.
For example alerts-loose-20150410.143637.csv
- Quotes are used within the files to handle occurrences of the delimiter. Quote characters in cells are enclosed in a second quote character. Null values from the database are written as \N.

Usage Tips

The following tips can help you plan your archiving strategy:

- We recommend running the archiver tool outside core operational hours to minimize the impact to users. Users of the interface should refresh their sessions after the utility has been used to delete data.
- Archiving often in small quantities allows for fast execution and minimal impact.
- You can set up a cron job to run the archiver daily, outside core operational hours.
- You can use a specific alert or Situation filter to remove targeted events.
- Exporting and/or removing large amounts of data on a running system can be slow.
- Exporting from a remote machine is slower because of network latency.
- The archiver tool can export data from the prc_earliest_highest_severity_event table but it cannot delete this data.
- To run the archiver tool remotely from Elasticsearch, follow the instructions in the [Distributed Installation](#) section of the Implementor Guide to configure Elasticsearch to listen on the external interface.
- You do not need to re-run the indexer after using the archiver tool to delete data. The -r option deletes records from Elasticsearch to keep the search feature synchronized with the database.

Archiver Command Reference

This is a reference for the archiving of Situations and alerts. The moog-archiver command line utility accepts the following arguments:

Argument	Input	Description
-a,--alert_filter <arg>	String: <filter name>	Include all loose alerts that match the specified global alert filter. Does not apply to alerts within Situations that are being archived.
-b,--update_batch_size <arg>	Integer: <number of alerts/Situation rows>	Defaults to 1000. Maximum number of alert /Situation rows to process at once during the export/deletion process. Increasing this value can speed up archiving but places more load on the database.
-d,--delimiter <arg>	String	Defaults to comma ",". Delimiter to insert between values in the export file.
-e,--export	-	Export the data to a file.

-f,--file_age <arg>	Integer: <number of days>	Delete files from the default directory /usr/local/archived that are older than the specified number of days.
-g,--loglevel <arg>	DEBUG INFO WARN FATAL	Defaults to WARN. Specify the output verbosity.
-h,--help	-	Display the moog-archiver utility syntax and option descriptions.
-i,--situation_filter <arg>	-	Include all Situations that match the specified global alert filter.
-l,--loose_alert_age <arg>	Integer: <number of days>	Export data related to loose alerts older than the specified number of days. Defaults to 28 for a standard database or 395 if the Historic Database is enabled.
-m,--include_statistics	-	Include the deletion of statistical data. Statistical data can only be deleted, not archived. Deletes statistical data from the active database whether database split is enabled or disabled.
-n,--statistics_age <arg>	Integer: <number of days>	Delete statistical data older than the specified number of days. Defaults to 28 for a standard database or 395 if the Historic Database is enabled.
-o,--retain_open	-	Only include data from Situations with Closed, Dormant or Superseded status. Cannot be used with the Situation filter -i
-p,--archive_path <arg>	String: <path>	Defaults to /usr/local/archived. Destination path for the archived data.
-r,--remove	-	Delete data from the database.
-s,--situation_age <arg>	Integer: <number of days>	Include Situation data (and alerts within Situations) older than the specified number of days. Defaults to 28 for a standard database or 395 if the Historic Database is enabled.
-t,--loose_alerts_only	-	Include loose alerts only. Cannot be used with -i -o -s
-y,--delay_time <arg>	Integer: <number of milliseconds>	Length of the delay (in milliseconds) between each batch operation. Can be used to slow the speed of archiving to reduce load on the database. Defaults to 0.
-z,--id_batch_size <arg>	Integer: <number of rows>	Maximum number of rows to process per batch during the export/deletion process. Increasing this value can speed up archiving but places more load on the database. Defaults to 100.

Authentication

You can configure different login authentication and security methods with Cisco Crosswork Situation Manager. For

- more information see: [Security Configuration](#)
- [Configure Single Sign-On with SAML](#).

SAML 2.0

Redirection Notice

This page will redirect to <https://docs.moogsoft.com/display/070000/Configure+Single+Sign-On+with+SAML>.

Security Configuration

- Introduction
- security.conf
- DB type
- LDAP type

Introduction

Lightweight Directory Access Protocol (LDAP) is an application protocol used for accessing a user directory in a network.

The LDAP security components of Cisco Crosswork Situation Manager are configured in the file called security.conf. You will also need information from your LDAP SME.

From LDAP

LDAP Critical Step	Description
The LDAP URL connection string	ex. "URL": "ldap://LDAPServerName:389"
Identify the User in the LDAP Tree that will have read only rights to search for end users of the tool	userDnLookupUser and userDnLookupPassword
User DN	The Distinguished Name is the unique path to any object in the active directory
"usernameAttribute": "sAMAccountName"	The attribute for the username
"userBaseDn" : "dc=corp,dc=standard,dc=com"	
"userDnLookupUser": "cn=LDAP-Moogsoft_User,ou=Accounts,ou=Users,dc=corp,dc=company,dc=com"	
"userDnLookupPassword": "PASSWORD"	Have the LDAP SME test this password and ensure the password is valid. Most service accounts have disabled the ability to logon which makes it challenging to properly test the password.
groupBaseDn": "OU=Groups,OU=SFG Users,DC=corp,DC=standard,DC=com"	Note: This is can be different than the location of the userDnLookupUser
"memberAttribute": "member"	
"groupNameAttribute": "cn"	

From Cisco

What to do	
Edit the Security.conf with all data provided by the LDAP SME as well as your Cisco Crosswork Situation Manager roles	"/usr/share/moogsoft/config/security.conf"
Ldapsearch Utility - It is highly recommended to use this tool as part of your LDAP configuration steps. The tool will enable you to test the existence location of the user being used to search LDAP for Cisco Crosswork Situation Manager end users	
ldapsearch -h LDAPServerName -u CN=LDAP-Moogsoft_User,OU=Accounts,OU=Users,DC=corp,DC=company,DC=com -x -W -b CN=LDAP-Moogsoft_DEV,OU=Groups,OU=Users,DC=corp,DC=company,DC=com	
service moogfarmd stop/start	After any change to the Security.conf file
service apache-tomcat stop/start	After any change to the Security.conf file

Teams Mapping

Team	Description
------	-------------

Role Map ie.."Cisco Crosswork Situation Manager Super User": "Super User"	This is just an example map as many roles as needed to properly configure Cisco Crosswork Situation Manager
Role Map ie.."Cisco Crosswork Situation Manager Operators": "Operator"	This is just an example map as many roles as needed to properly configure Cisco Crosswork Situation Manager to map to your groups or teams

Testing and Validation

What to do
The use of the ldapsearch utility will enable you to validate that your configuration settings and user settings are correct
You must engage the LDAP SME for validation of users and configuration settings
Using both ldapsearch and the LDAP SME are the key to a successful LDAP integration
When everything is configured you must stop and restart both moogfarmd and apache-tomcat
The two key logs to check once testing the configuration begins are: catalina.out = /usr/share/apache-tomcat/logs/catalina.out moogfarmd.log /var/log/moogsoft/moogfarmd.log

security.conf

\$MOOGSOFT_HOME/config/security.conf

This file allows different authentication and authorization sources to be defined. Each authentication/authorization source is called a Realm and there are three types:

Realm Type	Description
DB	A Cisco Crosswork Situation Manager database
LDAP	An external LDAP server

By default all the Realms within this file are commented out which means the default Moogsoft DB will be used. Edits to this file will require a restart of apache-tomcat for the changes to take effect. If any realms are defined in this file, the default DB Realm will not be used. This leads to the following configuration options:

Option	Description
1	All realms commented out will mean the moogsoft db is used. Only local db users will be able to log into the system
2	One or more LDAP realms (but not DB realm). Only LDAP users will be able to log into the system
3	One or more LDAP realms plus a DB realm (there can only be one db realm). Both LDAP users and local db users will be able to log into the system

If a user is defined in more than one Realm (with or without the same password) they will be authenticated and the login will succeed as long as at least one of the Realms authentication's succeeds.

Depending upon the type of Realm there may be additional configuration items required. The DB Realm does not require additional items but the LDAP one does. If both types are used, the DB Realm must come first in the file. All Realms are defined as follows (the name can be any you choose):

```
"Realm Name" : {  
    "realmType": "DB or LDAP",  
    ...additional json config ...  
}
```

DB type

An example DB type is given below:

```
{  
    #  
    # Realm type (DB or LDAP) #  
    "realmType": "DB"  
}
```

LDAP type

An example JSON config with all parameters is given below (each field is described in detail below):

```
{  
    "LDAP realm" : {  
        "realmType": "LDAP",  
  
        #  
        # [LDAP connection section] #  
  
        #  
        # The url indicates the protocol (ldap, ldaps) as well as the # host and port. This field is  
        mandatory.  
        #  
        #  
        #  
  
        #  
        # If predefinedUser is true then the user account information # must exist in the local DB  
        # (as well as the LDAP server).  
        #  
        # If predefinedUser is false then the LDAP information will # be used to create/update  
        # the user account.  
        #  
        # This field is optional (default value is false) #  
        "predefinedUser": false,  
  
        #
```

```

# [Authentication bind and attribute search section] #

#
# User DN (Distinguished Name) resolution for bind #
# There are two alternative resolution methods available to determine the
# user DN. The value of 'resolutionType' controls this and can
have one
# of the following values:
#
# "direct" - DN created based on userDnPostfix and
usernameAttribute. #
# REQUIRED FIELDS:
# "usernameAttribute" #
"userDnPostfix"
#
# e.g. #
# For the following config:
#
# "resolutionType" : "direct", # "direct" : {
#     "usernameAttribute": "uid",
#     "userDnPostfix": "ou=People,dc=moogsoft,dc=com" # },
#
# IF the user typed in a username of 'moo' for the above config
then the
# following DN would be created:
#
# "uid=moo,ou=People,dc=moogsoft,dc=com" #
#
# "lookup" - DN looked up based on userBaseSearchFilter and
usernameAttribute. #
# REQUIRED FIELDS:
# "usernameAttribute" #
"userBaseDn"
# "userBaseSearchFilter" #
# OPTIONAL FIELDS:
# "userDnLookupUser" (if not defined systemUser will be used) #
"userDnLookupPassword"
#
# e.g. #
# For the following config: #

```

```

# "resolutionType" : "lookup", # "lookup" : {
#     "usernameAttribute": "sAMAccountName",
#     "userBaseDn": "ou=People,dc=moogsoft,dc=com", #
#         "userBaseSearchFilter": "(objectclass=people)" # },
#
# If the user typed in a username of 'moo' for the above config

# following filter would be used (with the userBaseDn): #
# (&(objectclass=people)(sAMAccountName=moo)) #
# Note: If lookup matched more than one DN then the authentication

# fail. #

# "userDnResolution" : {
#     "resolutionType" : "direct", #
#         "direct" : {
#
#             "usernameAttribute": "uid",
#             "userDnPostfix": "ou=People,dc=moogsoft,dc=com" # }
#
# }, #
# "userDnResolution" : { "resolutionType" :
#     "lookup", "lookup" : {
#
#         "usernameAttribute": "sAMAccountName", "userBaseDn" :
#             "ou=People,dc=moogsoft,dc=com", "userBaseSearchFilter" :
#                 "(objectclass=person)",
#                 "userDnLookupUser": "uid=anon,ou=People,dc=moogsoft,
dc=com",
#
#         "userDnLookupPassword": ""
#
#     }
#
# },
#


#
# The attribute filter will be used in conjunction with the # user DN as base to retrieve
all user attributes.
#
# This field is optional (default '(objectclass=*)') #
"attributeSearchFilter": "(objectclass=*)",

#
# The attribute map defines a mapping from user DB fields (in the # database) to LDAP
attributes. The left hand key is the user DB # field name and the right hand side is the LDAP
attribute.
#
# Note: An LDAP attribute could potentially map to multiple user DB

```

```
# fields but a DB field can only map to one LDAP attribute. #
# Any LDAP attributes not in this list will be ignored. #
# This field is optional if predefinedUser is true otherwise it is # mandatory.
#
"attributeMap": { "fullname":
    "cn",
    "email": "mail"
},

#
# [LDAP Group search section] #

# An optional system user and password will be used to bind # and search for user group
information. If not provided
# then the user DN defined above will be used. #
# These fields are optional if predefinedUser is true OR the search for
# groups will be done under the user dn depending upon LDAP permissions).
#
"systemUser": "uid=appauth,ou=auth,ou=moogsoft,ou=Applications, dc=moogsoft,dc=com",
"systemPassword": "appauth",

#
# A filter (see below) will be used in conjunction with # the group base DN to
search for LDAP groups.
#
# Group filter ::= "(" <memberAttribute> "=" <UserDN> ")" #
# The groupBaseDn is optional if predefinedUser is true, otherwise
it is      # mandatory. The memberAttribute is optional (default 'member'). #
"groupBaseDn": "ou=groups,ou=moogsoft,ou=Applications,dc=moogsoft,
dc=com",
"memberAttribute": "member",

#
# Once the groups have been found using the above DN and filter # the group name attribute
will then be used to get the
# group name. The group name is then used to map the group # to a role using the role
map.
#
```


available:

```

# # 1. No SSL           - SSL configuration is not specified
#                               #
#                               #
#                               #
# ## 2. Express SSL - This is where SSL configuration is specified,
but      #                               #
# ##                               empty or only the SSL protocol is set and
specific   #                               #
# ##                               certificates do not need to specified.#
#                               #
# ## 3. Custom SSL          - This is where all the SSL configuration and    #
# ##                               certificates needed are specified to enable secure      #
# ##                               and authorised
communication.                                #
#                               #
# ## Note that Client key and certificate are optional.#
# ##                               If neither of those are specified, then client  #
# ##                               certification verification will not be performed.#
# ######
#                               #
# "ssl" :
# {
#     #
#     # Specify the SSL Protocol to use.
#     # If the configuration is not specified, "TLSv1.2" will be
used
#         # by default.
#         # JRE 8 supports "TLSv1.2", "TLSv1.1", "TLSv1", "SSLv3" #
#         "ssl_protocol" : "TLSv1.2", #
#         #
#         # The location of the SSL certificate, key files. #
#         # Relative pathing can be used, i.e. '.' to mean current
directory,
#         # './server.pem' or '../server.pem' etc. If neither
relative is
#         # nor absolute (using '/') path is used then $MOOGSOFT_HOME #      # prepended to
it.
#         # i.e. "config/server.pem" becomes "$MOOGSOFT_HOME/config
/server.pem"

```

```

#      #
#
#      #
# Specify the server certificate. #
# "server_cert_file" : "server.pem",
#
#      #
# Enable client authentication by specifying the client
#
#      #
# and key files below.
# The key file has to be in PKCS#8 format. #
# "client_cert_file" : "client.pem",
# "client_key_file"      : "client.key"
}#
}

}

```

realmType

The type of Realm (authentication/authorisation source). This can be either 'DB' or 'LDAP'.

url

A url as follows:

```

ldap://ip:port
ldaps://ip:port

```

The IP should be the IP address of the LDAP server and the port will typically be 389 (LDAP) and 636 (LDAPS).

By default port 389 is non-SSL, but SSL can be enabled, by upgrading the connection to secure with Start TLS - see [Start TLS](#). Note that Start TLS is recommended way to get secure connection with SSL, instead of using LDAPS.

If using LDAPS SSL then you will need to import the certificate into JAVA_HOME:

<http://docs.oracle.com/javase/tutorial/jndi/ldap/ssl.html>

The above link has a section on client requirements:

Run the following commands as Root

1. Enter the following:

```
# cd JAVA_HOME/lib/security (for JRE ..../jre/lib/security)
```

2. Check for the cacerts file in:

jre/lib/security

1. If this file exists, make a copy of it in the same directory and name it: jssecacerts
This is because if both cacerts and jssecacerts exist, jssecacerts will be used exclusively (so it is best to include the default certificates in jssecacerts by copying it).
3. Run the keytool command:

```
# keytool -import -file server_cert.cer -keystore jssecacerts
```

1. Java's default cacerts password is "changeit" (for the keytool)
2. If you encounter problems after importing the certificate then check:
 1. That you are importing to the correct JVM (The JVM that is running tomcat).
 2. Check to see if there is a certificate chain that needs importing as sometimes the whole chain needs to be imported.
3. The 'Start TLS Extension' is not currently supported (i.e. using ldap:// rather than ldaps:// for a TLS connection):
 1. <http://docs.oracle.com/javase/jndi/tutorial/ldap/ext/starttls.html>
 2. <http://www.ietf.org/rfc/rfc2830.txt>

predefinedUser

If predefinedUser is true then the user account information must exist in the local DB (as well as the LDAP server). To provision pre-existing users the stored procedure create_remote_user should be used e.g.

```
call create_remote_user ('ldapUser','Ldap User', 'All', 'Support', 'Super User', 'Europe/London', -1,  
'ldap@moogsoft.com', '@ldap', '555-8765');
```

If predefinedUser is false then the LDAP information will be used to create/update the user account (and the local DB user does not need to exist before logging in).

userDnResolution and resolutionType

A block of config that determines the mechanism used for obtaining the user DN that will be used in the LDAP bind. Within the userDnResolution block resolutionType is mandatory and should have one of two possible values (direct or lookup). Whatever the value there must also be a config block of the same name e.g.

```
"userDnResolution" : { "resolutionType" :  
    "direct", "direct" : { ... }  
}  
-or- "userDnResolution" : {  
    "resolutionType" : "lookup", "lookup" : {  
        ... }  
}
```

direct :

If resolutionType is direct then a user DN will be ‘built’ using the config fields usernameAttribute and userDnPostFix.

Example (“resolutionType” : direct):

```
DN ::= <usernameAttribute> “=” typedUsername ”,” <userDnPostfix>
```

e.g.

“usernameAttribute” = “uid”

“userDnPostfix” = “ou=People,dc=moogsoft,dc=com” Typed username
= fakeuser

Would lead to the following DN:

uid=fakeuser,ou=People,dc=moogsoft,dc=com

lookup :

If resolutionType is lookup then a user DN will be searched for in the LDAP server using the usernameAttribute as a filter and userBaseDn as a base to find the DN. The filter used for the user DN search is a combination (using AND) of userBaseSearchFilter and the usernameAttribute. If the user DN search returned more than one match then the login attempt would fail.

The actual user DN search will either use a specific userDnLookupUser or the systemUser (so at least one of these need to be configured).

Example (“resolutionType” : lookup):

```
User_DN_Filter ::= “(&(“ <userBaseSearchFilter> “)(“ <usernameAttribute> “=” typedUsername “))”
```

e.g.

“userBaseSearchFilter” = “objectclass=Person”

“usernameAttribute” = “sAMAccountName”

Typed username = fakeuser

Would lead to the following user DN search filter:

(&(objectclass=Person)(sAMAccountName=fakeuser)

Note: If no userBaseSearchFilter was specified then the filter would end up as follows:

(sAMAccountName=fakeuser)

usernameAttribute

Depending on the value for userDnLookup this key field is used to either build or search for the user DN that is used in the LDAP bind. See userDnResolution section for more details.

userDnPostFix

Used in conjunction with the usernameAttribute to build the user DN (Distinguished Name) that is used for authentication. See userDnResolution section for more details.

userBaseDn

When resolutionType is 'lookup' this field will be used to determine the base subtree of the LDAP structure of which to do the user DN search.

e.g.

```
"userBaseDn" : "ou=People,dc=moogsoft,dc=com"
```

userDnLookupUser and userDnLookupPassword

When resolutionType is 'direct' the userDnLookupUser and userDnLookupPassword if defined will be used to search for the user DN (based on the entered username). If these fields are not defined then the LDAP module will attempt to use the systemUser and systemPassword instead.

If the userDnLookupPassword is not defined (or is empty) the LDAP module will attempt an unauthenticated user bind (Special configuration would be needed in the LDAP server to allow this)

encryptedUserDnLookupPassword

If you want to encrypt your passwords using the Moog Encryptor, comment out the userDnLookupUser and uncomment encryptedUserDnLookupPassword.

```
"userDnResolution" : {
    "resolutionType" : "lookup", "lookup" : {
        "usernameAttribute": "sAMAccountName", "userBaseDn" :
        "ou=People,dc=moogsoft,dc=com", "userBaseSearchFilter" :
        "(objectclass=person)",
        "userDnLookupUser": "uid=anon,ou=People,dc=moogsoft,
        dc=com",
        "#" "userDnLookupPassword": ""#
        #for encrypting passwords via moog_encrypter comment the
        userDnLookupPassword, and uncomment encryptedUserDnLookupPassword
        "encryptedUserDnLookupPassword": ""
    }
},
```

attributeSearchFilter

An LDAP filter that is used for searching for user attributes. This filter will be passed with the user DN as base to find all user attributes. If not specified this will be given the following default value:

```
(objectclass=*)
```

attributeMap

After searching LDAP for user attributes this configuration will be used to set certain user fields (for any mapped LDAP attributes). The format is as follows:

```
"attributeMap": {  
    "db_column_5": "ldap_attribute_1",  
    "db_column_2": "ldap_attribute_8",  
    "db_column_3": "ldap_attribute_8",  
}
```

The original version of attributeMap had the attributes swapped around so that the LDAP attribute was on the left and the DB column name was on the right. This had serious limitations in that it means an LDAP attribute could only map to a single DB column. Since it is useful to have a single LDAP attribute map to multiple DB columns this was swapped around

systemUser and systemPassword

With the LDAP protocol, the system will send two bind and two search requests. If no system user is configured then the user bind (used for authentication) will be re-used for the LDAP group search also.

groupBaseDn

A DB (Distinguished Name) for the part of the LDAP structure that contains the user groups. This will be used in conjunction with the member Attribute to find any LDAP groups the user belongs to. These groups are then mapped to a local role using the roleMap.

memberAttribute

The member attribute is used to build a group filter to find groups.

```
Filter ::= "(" <memberAttribute> "=" <UserDN> ")"
```

groupNameAttribute

This is the name of the LDAP attribute (for the LDAP group) that identifies the group name. This group name will then be used by the roleMap.

roleMap

Users are linked to Roles via LDAP groups. Users are members of LDAP groups and these groups are used to assign Users Roles. The format is as follows:

```
"roleMap": {  
    "role-admin": "Super User", "role-other": "Customer"  
}
```

The left hand side of the above mapping is the LDAP group and the right hand side is the Cisco Crosswork Situation Manager role name (as defined in the roles table). The LDAP group names (as defined by the groupNameAttribute e.g. CN) will be mapped to a role using this configuration.

Users who are not members of any of the mapped LDAP groups will not be able to log in.

assignTeams

Users can be assigned to a team via LDAP groups. Users are members of LDAP groups and these groups are used to assign Users Teams. To add this optional feature add the assignTeams entry to enable it. Please note, if this is enabled, any user assignment to a team done by the UI might get overwritten. This entry should have the following members:

teamMap

A map from the group name in LDAP to the team name in Cisco Crosswork Situation Manager.

useGroupName

Optional Boolean argument. If set to true, and the group name cannot be found in the teamMap (see above) or the roleMap (see above) it will assign the user to a team with the same name as the LDAP group.

createNewTeams

Optional Boolean argument. If set to true, and Cisco Crosswork Situation Manager cannot find an existing team with the wanted name (either in the teamMap or when useGroupName is set) it will create a new team. Please note, this team will be created without services, situations filter or alerts filter. This will need to be done using the UI or graze.

Example for the assignTeams format:

```
assignTeams :{
    teamMap: {
        "ldap-group-name": "Team name",
        "yet_another_group": "Another team"
    },
    useGroupName : true,
    createNewTeams: false
}
```

Start TLS

ssl

SSL can be enabled for non-SSL connections (by default on port 389), by upgrading the connection to be secure with SSL using Start TLS.

This can be achieved by enabling "ssl" configuration section for LDAP Realms. Once enabled, Express or Custom mode can be used.

In Express mode, none of the certificates or keys needs to be specified. Secure connection will be created, even though the certificate is not validated.

In Custom mode, the provided certificates and keys needs to be valid and match the LDAP Server certificate.

Additional client authentication can be provided with both *client_cert_file* and *client_key_file* specified, where the *client_key_file* needs to be issued by this same CA as the *server_cert_file*.

Note that client authentication needs to be explicitly enabled in LDAP Server schema.

ssl_protocol

(Optional) Specifies with SSL Protocol to use.

With JRE 8, it can be any of those "TLSv1.2", "TLSv1.1", "TLSv1", "SSLv3".

By default that is "TLSv1.2".

The location of SSL certificate, key files:

Relative pathing can be used, i.e. '.' to mean current directory, './server.pem' or '../server.pem' etc. If neither relative nor absolute (using '/') path is used then \$MOOGSOFT_HOME is prepended to it. i.e. "config/server.pem" becomes "\$MOOGSOFT_HOME/config/server.pem"

server_cert_file

Path to the LDAP server certificate file.

client_cert_file

Path to the Client certificate file.

client_key_file

Path to the Client private key file. The provided key needs to be in PKCS#8 format.

Configure Single Sign-On with SAML

You can allow your Cisco Crosswork Situation Manager users to log in with Single Sign-On (SSO) if you configure Security Assertion Markup Language (SAML) realm.

When you enable the SAML integration, your SAML identity provider (IdP) can exchange authorization and authentication data securely with your service provider (SP), Cisco Crosswork Situation Manager. The integration redirects you from Cisco Crosswork Situation Manager' standard login page to the IdP's login page. You can log in to Cisco Crosswork Situation Manager if you provide the IdP with valid authentication details.

Cisco Crosswork Situation Manager implements SAML 2.0 using the SAML v3 Open Library. SAML 2.0 supports

- the following bindings: HTTP-Artifact
- HTTP-POST
- HTTP-POST-SimpleSign
- HTTP-Redirect
- SOAP

See [Open SAML v3](#) for more information.

Before You Begin

Before you start to set up SAML, ensure you have met the following requirements:

- You have an active SAML Identity Provider account with administrator privileges.
- Ensure the web host URL in \$MOOGSOFT_HOME/config/servlets.conf is the same as your Cisco Crosswork Situation Manager instance URL:

```
# The web host address (default to https://localhost) webhost :  
"https://example.moogsoftaiops.com",
```

Configure SAML Identity Provider

You can configure your IdP to integrate with Cisco Crosswork Situation Manager and enable SSO. Refer to your IdP's documentation for instructions. Configuration differs for each IdP but common settings include:

- **SSO URL:** The Cisco Crosswork Situation Manager URL that sends a SAML login request to the IdP:

```
https://example.moogsoftaiops.com/moogsvr/mooms?request=samlRequest
```

- **Assertion Consumer Service URL:** The Cisco Crosswork Situation Manager URL that receives the IdP response to each SAML assertion:

```
https://example.moogsoftaiops.com/moogsvr/mooms?request=samlResponse
```

- **Entity ID:** A unique identifier for the SP SAML entity:

```
"https://example.moogsoftaiops.com/moogsvr/mooms"
```

After you complete the IdP configuration, it generates an IdP metadata file in .xml format. Some IdPs also allow you to generate an X509 self-signed certificate. Save the certificate and add it to your SP metadata file if you want your IdP to encrypt SAML assertions.

Copy the Identity Provider Metadata File

You create the IdP metadata file as part of the IdP configuration. This .xml file provides Cisco Crosswork Situation Manager with a security certificate, endpoints and other processing requirements.

To add this file to your SAML configuration:

1. Save the IdP metadata file to your local machine.
2. Copy the metadata file to \$MOOGSOFT_HOME/usr/share/moogsoft/etc/saml.
3. Grant the Apache Tomcat user read permissions to the metadata file. For example:

```
chmod 644 my_idp_metadata.xml
```

Create the Service Provider Metadata File

You must create an SP metadata file and send it to the IdP you want to integrate with Cisco Crosswork Situation Manager.

Some IdPs offer an SP metadata generator. If your IdP does not generate the SP metadata file, you can create one manually. See [Build a Service Provider Metadata File](#) for information.

After you have generated your SP metadata file:

1. Copy the file to \$MOOGSOFT_HOME/etc/saml.
2. Grant the Apache Tomcat user read permissions to the metadata file. For example:

```
chmod 644 my_sp_metadata.xml
```

Configure the SAML Realm

You enable SAML authentication in Cisco Crosswork Situation Manager by creating and configuring a SAML realm. You can only configure and use one SAML Realm at a time. See [Security Configuration Reference](#) for full descriptions of the available properties.

To configure your SAML realm:

- 1 Uncomment the "my_saml_realm" section in the \$MOOGSOFT_HOME/config/security.conf configuration file. Rename the realm to meet your requirements.
- 2 Configure the locations of your metadata files:
 - **idpMetadataFile:** Location of the identity provider's metadata file.
 - **spMetadataFile:** Location of the service provider's metadata file.
- 3 Optionally configure the roles, teams and primary group mappings for new users that log in to Cisco Crosswork Situation Manager using SAML:
 - **defaultRoles:** Default roles that Cisco Crosswork Situation Manager assigns to new users at first login. **defaultTeams:** Default teams that Cisco Crosswork Situation Manager assigns to new users at first login. **defaultGroup:** Default primary group that Cisco Crosswork Situation Manager assigns to new users at first login.

- 4 Configure the mappings for existing users that log in to Cisco Crosswork Situation Manager using SAML. You can choose either username or email:

- **existingUserMappingField**: Defines the field that Cisco Crosswork Situation Manager uses to map existing users to your IdP users.
- 5 Configure the mapping of the IdP's provided attributes:
- **username**: Defines the IdP user attribute that maps to username in Cisco Crosswork Situation Manager.
 - **email**: Defines the IdP user attribute that maps to email in Cisco Crosswork Situation Manager.
 - **fullname**: Defines the IdP user attribute that maps to full name in Cisco Crosswork Situation Manager.
- 6 Optionally configure your keystore and private key passwords if you want to use encryption with SAML. See [Optional SAML Security Features](#):
- **keystorePassword**: Your keystore password.
 - **privateKeyPassword**: Your private key password.
- 7 Optionally configure the lifetime of each SAML assertion. See [Optional SAML Security Features](#):
- **maximumAuthenticationLifeTime**: Maximum time in seconds for Cisco Crosswork Situation Manager to receive an IdP's SAML assertion before it becomes invalid.
- 8 Optionally configure the Service Provider Entity Id. See [Optional SAML Security Features](#):
- **serviceProviderEntityId**: Service Provider Entity ID assertion number.
- 9 Restart the Apache Tomcat service:

```
service apache-tomcat restart
```

Enable Encrypted Assertion

To enable encrypted assertion for SAML with Cisco Crosswork Situation Manager:

1. Copy the location of your KeyStore file. This defaults to \$MOOGSOFT_HOME/etc/saml/<name of realm>_keystore. Cisco Crosswork Situation Manager generates this file when you create the realm.
2. Log in to your SAML IdP and enable encrypted assertions. Refer to your IdP's documentation for information.
3. Provide your KeyStore password and import your KeyStore file if required to do so.

Once enabled, the IdP encrypts all SAML assertions made with Cisco Crosswork Situation Manager.

Set an Assertion Time Limit

You can set the assertion time limit for Cisco Crosswork Situation Manager. The assertion time limit is the duration between the IdP providing the SAML assertion and when Cisco Crosswork Situation Manager accepts it.

Cisco Crosswork Situation Manager accepts a delay of up to an hour by default. You can specify a different time to meet your requirements.

```
# The time in seconds that a valid SAML assertion has from being issued # by the IdP to being received
# by Cisco Crosswork Situation Manager before that assertion # is considered invalid. Default is 1 hour.
,"maximumAuthenticationLifetime": 3600
```

Enable Entity ID Assertion

You can enable entity ID assertion, also known as audience restriction, to restrict SAML assertions to Cisco Crosswork Situation Manager.

You configure the unique SP entity ID in \$MOOGSOFT_HOME/config/security.conf. You must also configure this in your IdP. The values must match for successful SAML authorization:

```
# Service Provider Entity Id.  
# Some IdPs require SP Entity ID assertion, this can be configured here.  
, "serviceProviderEntityId": "MoogsoftAIOps"
```

Map User Attributes

When you create your realm, you can configure the attributes your Identity Provider passes to Cisco Crosswork Situation Manager at SAML authentication.

By default, the IdP email attribute maps to both the Cisco Crosswork Situation Manager username and email. The Cisco Crosswork Situation Manager full name maps to First Name and Last Name from the IdP:

```
"username": "$Email", "email":  
"$Email",  
"fullname": "$FirstName.$LastName"
```

You may see errors indicating failure to configure an attribute mapping or indicating the IdP's failure to provide a configured attribute if something goes wrong at login.

You can map other IdP user attributes such contact number, department, primary group and time zone:

```
"contactNumber" : "phone", "department"  
: "department", "primaryGroup" :  
"primaryGroup", "timezone" : "timezone",
```

If you already have users in Cisco Crosswork Situation Manager you can map the user attributes to the IdP using the existingUserMappingField:

```
"existingUserMappingField": "username",
```

When a user logs in via the IdP for the first time but does not map with an existing user entry, Cisco Crosswork Situation Manager creates a new user.

You can define which primary group, roles and teams to assign users to using the defaultRoles, defaultTeamsand defaultGroup properties in the SAML realm configuration.

You can map the IdP's attribute for team names using teamAttribute. IdP team names map to Cisco Crosswork Situation Manager primary groups by default. You can configure which IdP attribute maps to Cisco Crosswork Situation Manager team names using teamMap:

```
"assignTeams" : {  
    # SAML attribute containing the team names. Defaults to "groups" "teamAttribute" : "groups",  
    # Mapping of SAML group name or custom attribute to Cisco Crosswork Situation Manager  
    teams  
        "teamMap" : {  
            "IdP Team" : "Cisco Crosswork Situation Manager Team",  
            "Another IdP Team" : "Another Cisco Crosswork Situation  
            Manager team"
```

To create a team that does not exist already, enable the createNewTeamsproperty:

```
"createNewTeams" : true
```

If you enable createNewTeams, Cisco Crosswork Situation Manager assigns users to the teams it creates as part of the SAML login instead of the default SAML teams.

You can map the IdP attribute for roles using roleAttribute. You can map the IdP roles to Cisco Crosswork Situation Manager roles using roleMap:

```

"assignRoles" : {
    # IdP attribute containing role information. If this is the same as "teamAttribute",
    # Cisco Crosswork Situation Manager will checks any groups against roles first
    "roleAttribute" : "groups",
    # Map SAML roles to Cisco Crosswork
    Situation Manager roles "roleMap" : {
        "IdP Standard User" : "Operator", "IdP Manager
        User" : "Manager"
    }
}

```

Configure SAML Logout URL

After you enable SAML, you can configure a different logout page to display when a Cisco Crosswork Situation Manager user ends their session. To configure a different logout page:

- 1 Edit the \$MOOGSOFT_HOME/ui/web.conf configuration file:

```

"authentication": { "pages": [
    {
        "login": "/login/", "logout": "/logout/",
        "failedLogin": "/login/?error=true", "sessionTimeout": "/logout/?error=session",
        "dbFailure": "/login/?error=dbfailure"
    },
    "paramNames": { "userId": [
        "userid",
        "password": "password"
    ]}
],
}

```

- 2 Change the sub URL for "logout" to meet your requirements.
- 3 Save the changes.

After you have completed the change, Cisco Crosswork Situation Manager displays the new logout path when a session expires or if you log out.

Example SAML Realm

You can use the default SAML realm in \$MOOGSOFT_HOME/config/security.conf for reference:

```

"my_saml_realm" : {
    "realmType": "SAML2",
    "idpMetadataFile": "/usr/share/moogsoft/etc/saml/my_idp_metadata.
xml",
    "spMetadataFile": "/usr/share/moogsoft/etc/saml/my_sp_metadata.xml", "defaultRoles": [ "Operator"
],
    "defaultTeams": [ "Cloud DevOps" ], "defaultGroup":
"End-User", "existingUserMappingField": "username",
    "username": "$Email",
    "email": "$Email",
    "fullname": "$FirstName $LastName"
    "contactNumber" : "phoneNumber", "department"
: "dept", "primaryGroup" : "group", "timezone" :
"timezone", "assignTeams" : {
        "teamAttribute" : "groups", "teamMap"
        : {
            "Cloud Team" : "Cloud DevOps", "Database
Team" : "Databse DevOps"
        }
    },
    "createNewTeams" : true
},
    "assignRoles" : {
        "roleAttribute" : "groups", "roleMap" : {
            "Standard User" : "Operator", "Manager
User" : "Manager"
        }
    }
},
    "keystorePassword": "my_realm_secret"
    , "privateKeyPassword": "my_realm_secret"
    , "maximumAuthenticationLifetime": 60
    , "serviceProviderEntityId": "MoogsoftAIOps"
}

```

Build a Service Provider Metadata File

You must build a Service Provider (SP) metadata file in order to configure SAML-based Single Sign-On with Cisco Crosswork Situation Manager.

The SP metadata .xml file contains all of the keys, services and URLs defining the SAML endpoints. You can use your IdP's SP metadata file generator if it has one. If not you can create the file manually.

Build Your Metadata File

To manually create your SP metadata file:

1. Copy the .xml template from the code block:

```

<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="https://localhost/moogsvr/mooms"
    <md:SPSSODescriptor AuthnRequestsSigned="true" WantAssertionsSigned="true"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
        <md:KeyDescriptor>
            <ds:KeyInfo xmlns:ds="http://www.w3.org/2000
/09/xmldsig#">
                <ds:X509Data>
                    <ds:X509Certificate>
MIIC/jCC AeagAwIBAgIQCGehfcnv6r5My/fnrbfDejANBgkqhkiG9w0BAQsFADAV
MRMwEQYDVQQDEwp3d3cuc3AuY29tMB4XDTEzMTEyMjA4MjMyMVoXDTQ5MTIzM
TE0
MDAwMFowFTETMBEGA1UEAxMKd3d3LnNwLmNbTCCASIwDQYJKoZIhvcNAQEBB
QAD
ggEPADCCAQoCggEBAMPm/ew9jaGWpQS1C7KtpvgzV4nSOIFPgRt/nlRYR+pUWdDE
fSKmyjK28nkQ1KKujRJTnvnmZydmUrmEFpVv+giBiUkvCJY3PxZ/EDSsF3R/OzWh
kUv5nfAXPnqkX9x22b6+vUof6WiLGyAW6lOYMCVAdjTSI9pSaUtIaANdx9maERcT
9eQbGSnjim0WurFRYs9ZE8ttErrMH9+Su4246YDqOPAkz6La4cHHMPQdcFQT5p/c
uXBfU1vl1tWdB EgAY3xHYZE8u5TTJ/vp9UxyU1MwfeO2g9VDRcokLQHrj6wFxtvu
fA+WtUKYJGUu2p/qSuaw7eS6UFjUn49aVqg9OacCAwEAAaNKMEgwRgYDVR0BBD8w
PYAQ1/S0ibdvfdFkJ9T9oIPluKEXMBUxEzARBgNVBAMTCnd3dy5zcC5jb22CEAhn
oX3J7+q+TMv35623w3owDQYJKoZIhvcNAQELBQADggEBAAHlmVoAZUt6paeFvtQb
c/iaJe/Fhd+JG1U0jyjlFDcCn8erLihEbhb3mFBBMF25oO67gfA1JJXZrmHry3NI
OZuovqRqm8v7wg8n0nQa1HUWkUC2TBgfg1HE8/2rmSF2PngiEi18VOxRDxx0WXMN
ZX6JebJ1kCOCpT/x7aupS7T1GrIPmDLxjnC9Bet7pRynfomjP/6iU21/xOIF6xB9
Yf1a/kQbYdAVt2haYKIfvaF3xsq1X5tCXc9ijhBMgyaoqA+bQJD/l3S8+yCmMxEY
ZjAVLEkyGlU4Uwo01cKEYbXIG/YVq+4CaIRxIfMvV+j8gzTLHTXI+pHEMfMhyYa0
pzM=
                    </ds:X509Certificate>
                </ds:X509Data>
            </ds:KeyInfo>
        </md:KeyDescriptor>
        <md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="https://localhost:44360/SAML/SingleLogoutService"/>
        <md:AssertionConsumerService index="0" isDefault="true"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="https://localhost/moogsvr/mooms?request=samlResponse" index="0"/>
    </md:SPSSODescriptor>
</md:EntityDescriptor>

```

2. Configure the mandatory elements in the metadata file:

- **entityID**: Unique identifier or name for the SP. This should be a URL or a URN.
- **AssertionConsumerService**: URL or endpoint that receives SAML responses from the IdP.

3. Add the X509 self-signed certificate you create when you configure your IdP.

4. Configure the other elements to meet your requirements. See [Service Provider Metadata Reference](#) for full descriptions of the available elements.
5. Save the SP metadata file to a path on your local machine.

After you have created the metadata file, you must copy it to your Cisco Crosswork Situation Manager machine to continue with the SAML configuration. See [Configure Single Sign-On with SAML](#).

Service Provider Metadata Reference

This is a reference for [Build a Service Provider Metadata File](#). Each SP metadata .xml file accepts the following elements:

entityID: Unique identifier or name for the service provider. The ID should be a URN or a URL.

Type: String
Required: Yes
Example: <https://example.moogsoftaiops.com/moogsvr/mooms>

ID: Unique identifier for the root metadata element.

Type: String
Required: No
Example: TW9vZ3NvZnRBSU9wcw==

validUntil: Defines the expiration date of the metadata file. The date should be in ISO 8601 format.

Type: String
Required: No
Example: 2018-08-10T07:47:41+00:00

AuthnRequestsSigned: If enabled, Cisco Crosswork Situation Manager signs SAML authentication requests as part of the Single Sign-On.

Type: Boolean
Required: No
Default: false

WantAssertionsSigned: If enabled, Cisco Crosswork Situation Manager expects IdPs to sign any SAML assertions it sends.

Type: String
Required: No
Default: false

KeyDescriptor: Defines the type of signing or the type of encryption that Cisco Crosswork Situation Manager uses.

Type: String
Required: No
One of: use = "signing", use = "encryption"

X509Certificate: Self-signed certificate that allows Cisco Crosswork Situation Manager to sign and encrypt each SAML assertion. The certificate should be in DER format and base-64 encoded.

Type: String
Required: No
Example: MIIDijCCAAnICCQD[...]+6SBfDCrWFsw==

AssertionConsumerService: Defines the URL or endpoint that receives the SAML assertions. The Location is for the URL and the Binding identifies the method. Supported bindings include: HTTP-Artifact, HTTP-POST, HTTP-POST-SimpleSign, HTTP-Redirect and SOAP.

Type: String
Required: Yes
Example: Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST Location="<https://localhost/moogsvr/mooms?request=samlResponse>"

Security Configuration Reference

This is a reference for security configuration in Cisco Crosswork Situation Manager. You can edit \$MOOGSOFT_HOME/config/security.conf to configure security features such as SSL, LDAP and [SAML](#).

SAML Service Provider Properties

You can configure the SAML realm by giving it a name and changing the values of the following properties:

idpMetadataFile: Location of the identity provider's metadata file. The metadata file provides information on how to connect to the IdP. Cisco Crosswork Situation Manager requires the file to be in .xml format.

Type: String
Required: Yes
Default: "/usr/share/moogsoft/etc/saml/my_idp_metadata.xml"

spMetadataFile: Location of the service provider's metadata file. **Cisco Crosswork Situation Manager writes the SP metadata information to this file.** This location needs to be accessible and editable by the Apache Tomcat user. Cisco Crosswork Situation Manager requires the file to be in .xml format. If your IdP does not have an SP metadata file generator, you can create one manually. See [Build a Service Provider Metadata File](#) for instructions.

Type: String
Required: No
Default: "/usr/share/moogsoft/etc/saml/my_sp_metadata.xml"

defaultRoles: Default roles that Cisco Crosswork Situation Manager assigns to new users upon first login using SAML. If the user already has a role mapping, Cisco Crosswork Situation Manager uses that instead.

Type: Array
Required: Yes
Default: ["Operator"]

defaultTeams: Default teams that Cisco Crosswork Situation Manager assigns to new users upon first login using SAML. You can create an empty list if you do not want to assign new users to a team.

Type: Array
Required: No
Default: ["Cloud DevOps"]

defaultGroup: Default primary group that Cisco Crosswork Situation Manager assigns to new users upon first login using SAML.

Type: Array
Required: Yes
Default: ["End-User"]

SAML User Mapping Properties

You can configure how to map IdP user fields to existing Cisco Crosswork Situation Manager users and how to map user fields for new users. All mappings are case sensitive. Each mapping follows the format "MoogsoftAttribute" : "IdPAttribute".

existingUserMappingField: Defines the field that Cisco Crosswork Situation Manager uses to map existing users to your IdP users.

Type: String
Required: No
One of: username, email
Default: "username"

username: Defines the IdP's attribute that maps to username in Cisco Crosswork Situation Manager.

Type: String
Required: No
Default: "\$Email"

email: Defines the IdP's attribute that maps to email in Cisco Crosswork Situation Manager.

Type: String
Required: No
Default: "\$Email"

fullname: Defines the IdP attributes that map to full name in Cisco Crosswork Situation Manager.

Type: String
Required: No
Default: "\$FirstName \$LastName"

SAML Optional Properties

You can customize your SAML realm with a number of optional properties:

contactNumber: Defines the IdP attribute that maps to contact number in Cisco Crosswork Situation Manager.

Type: String
Required: No
Default: "phone",

department: Defines the IdP attribute that maps to department in Cisco Crosswork Situation Manager.

Type: String
Required: No
Default: "department",

primaryGroup: Defines the IdP attribute that maps to primary group in Cisco Crosswork Situation Manager.

Type: String
Required: No
Default: "primaryGroup",

timezone: Defines the IdP attribute that maps to time zone in Cisco Crosswork Situation Manager.

Type: String
Required: No
Default: "timezone",

teamAttribute: Defines the IdP attribute that maps to teams in Cisco Crosswork Situation Manager.

Type: String
Required: No
Default: "groups"

teamMap: Defines the IdP attribute or custom attribute that maps to team names in Cisco Crosswork Situation Manager.

Type: JSON Object
Required: No
Default: { "IdP Team" : "Cisco Crosswork Situation Manager Team", "Another IdP Team" : "Another Cisco Crosswork Situation Manager team" }

createNewTeams: Creates a team or teams if they did not exist in Cisco Crosswork Situation Manager already. If you left teamMap empty, the teams adopt their IdP teams names.

Type: Boolean
Required: No
Default: false

roleAttribute: Defines the IdP attribute containing role information.

Type: String
Required: No
Default: "groups"

roleMap: Defines the IdP attribute that maps to Cisco Crosswork Situation Manager roles.

Type: JSON Object
Required: No
Default: { "IdP Standard User" : "Operator", "IdP Manager User" : "Manager" }

keystorePassword: Your keystore password. Any whitespace in the name is replaced with an underscore.

Type: String
Required: No
Default: "<my_realm>_secret"

privateKeyPassword: Your private key password. Any whitespace in the name is replaced with an underscore.

Type: String
Required: No
Default: "<my_realm>_secret"

maximumAuthenticationLifetime: Maximum time in seconds for Cisco Crosswork Situation Manager to receive an IdP's SAML assertion before it becomes invalid.

Type: Integer
Required: No
Default: 2592000(720 hours)

serviceProviderEntityId: Service Provider Entity ID assertion number. Some IdPs require this ID.

Type: String
Required: No
Default: "MoogsoftAIOps"

Configure the Tool Runner

You need to add three values in the servlets.conf file to configure Tool Runner before you can use it in the UI.

The toolrunner uses ssh to login to a system with a certain username and password to run tools/integrations. As a result, 'PasswordAuthentication' must be set to 'yes' in /etc/ssh/sshd_config for this to work. Changes to that file require restarting the sshd service before they take effect.

Edit the servlets.conf file manually

From the Command Line:

1. Identify/create the user on your OS that will run Tools.
2. Edit the file:
\$MOOGSOFT_HOME/config/servlets.conf

Ensure you have the correct permissions in your System to edit this file

3. Find the 'Configuration for the toolrunner section' and edit three values in the 'toolrunner section':

Parameter	Description
toolrunnerhost	Default: localhost If running a distributed install this will be the hostname of the machine where apache-tomcat is installed
toolrunneruser	Default: moogtoolrunner The user named here needs to exist in your System and have appropriate permissions to run the needed tools (as identified in step 1 above)
toolrunnerpassword	Default: moogtoolrunner This needs to be the password of the user defined in toolrunneruser (as identified in step 1 above)

```
# Configuration for the toolrunner toolrunner :  
  
{  
  
    # The SSH timeout  
  
    sshtimeout: 900000,  
  
    # The toolrunner host to run on  
  
    toolrunnerhost: "localhost".  
  
    # The toolrunner user name  
  
    toolrunneruser: "moogtoolrunner".  
  
    # The toolrunner user password.  
  
    # Use either toolrunnerpassword or toolrunnernpassword.
```

```

toolrunnerpassword: "moogtoolrunner"

# encrypted_toolrunnerpassword:
"rmW2daCwMyI8JGZygfEJj0MZdbIkUqX3tT/OIVfMGyI=",

# Uncomment to overwrite the level of logging.

# loglevel: "WARN".

# Uncomment to overwrite the web host address

# webhost: "https://localhost"

# Uncomment to change the ha definitions # ha :

# {

#     cluster: "MOO",

#     instance: "toolrunner",

#     group: "toolrunner",

#     start_as_passive: false

# }

},

```

4. Stop and restart Apache.

```

service apache-tomcat stop service
apache-tomcat start

```

5. Stop and restart moogfarmd.

```

service moogfarmd stop service
moogfarmd start

```

The toolrunner is now available in the UI.

Example servlets.conf file

(\$MOOGSOFT_HOME/config/servlets.conf)

```
#####
# Copyright (c) Cisco Inc 2016 #
#
#-----# #
# The contents of this configuration file may be copied, #
# amended and used to create derivative works. #
#
#-----# #
#####


#-----#
# Servlets Configuration
#-----#
{



# The level of warning (ALL, INFO, WARN, DEBUG or NONE) # Default to
# WARN
loglevel: "WARN",



# The web host address (default to https://localhost) webhost : "https://freida",



# Configuration of the Moog Server moogsvr:
{



# Set to true to force each user to agree to the end user license # agreement
```

```
eula_per_user: false,  
  
# Location in the file system in which media files will be saved. # Default to /tmp  
  
cache_root: "/var/lib/moogsoft/moog-data",  
  
# Number of database connections (default to 10) db_connections:  
10,  
  
# Number of database connection in the priority queue (default to  
10)  
  
priority_db_connections: 10  
  
# Uncomment to overwrite the level of logging. # loglevel:  
"WARN",  
  
# Uncomment to overwrite the web host address # webhost:  
  
"https://localhost"  
  
# Uncomment to change the ha definitions # ha :  
  
# {  
#     cluster: "MOO",  
#     instance: "moogsvr", #group:  
"moogsvr",  
#     start_as_passive: false # }  
  
},  
  
# Configuration of the poller moogpoller :  
  
{  
  
# Uncomment to overwrite the level of logging. # loglevel:  
"WARN",
```

```
# Uncomment to overwrite the web host address # webhost:  
  
"https://localhost"  
  
# Uncomment to change the ha definitions # ha :  
  
# {  
  
#     cluster: "MOO",  
  
#     instance: "poller", #  
  
#         group: "poller",  
  
#         start_as_passive: false # }  
  
},  
  
# Configuration for the toolrunner  
  
# details in http://docs.moogsoft.com/display/060000/Tool+runner toolrunner :  
  
{  
  
# The SSH timeout  
  
sshtimeout: 900000,  
  
# The toolrunner host to run on  
  
toolrunnerhost: "localhost",  
  
# The toolrunner user name toolrunneruser:  
  
"moogtoolrunner", # The toolrunner user  
  
password.  
  
# Use either toolrunnerpassword or toolrunnerpassword. toolrunnerpassword:  
  
"moogtoolrunner"  
  
# encrypted_toolrunnerpassword: "rmW2daCwMyI8JGZygfEJj0MZdbIkUqX3tT  
/OIVfMGyI=",
```

Uncomment to overwrite the level of logging.

```

# loglevel: "WARN",

# Uncomment to overwrite the web host address # webhost:

"https://localhost"

# Uncomment to change the ha definitions # ha :

{

#     cluster: "MOO",

#     instance: "toolrunner", #

group: "toolrunner",

#     start_as_passive: false # }

},

# Graze configuration graze :

{

# Uncomment to overwrite the level of logging. # loglevel:

"WARN"

},

# Events configuration events

:

{

# Uncomment to overwrite the level of logging. # loglevel:

"WARN"

}

```

Edit the servlets.conf file automatically

1. Copy the following code into a file and name it, for example, toolrunner_script.sh

```
#!/bin/bash
# toolrunner script to automatically create a user that matches the default configuration
DIVD="
=====
=====
DONN="
=====
===== DONE"

echo $DIVD
echo "LOG: creating moogtoolrunner user" useradd -g
moogsoft moogtoolrunner
echo -e "moogtoolrunner\nmoogtoolrunner\n" | passwd moogtoolrunner echo "" >>
/home/moogtoolrunner/.bashrc
echo '# adding path to moogtoolrunner user' >> /home/moogtoolrunner/. bashrc
echo 'export MOOGSOFT_HOME=/usr/share/moogsoft' >> /home
/moogtoolrunner/.bashrc
echo 'export JAVA_HOME=/usr/java/latest' >> /home/moogtoolrunner/. bashrc
echo 'export APPSERVER_HOME=/usr/share/apache-tomcat' >> /home
/moogtoolrunner/.bashrc
echo 'export PATH=$PATH:$MOOGSOFT_HOME/bin:$MOOGSOFT_HOME/bin/utils'
>> /home/moogtoolrunner/.bashrc
echo 'export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$MOOGSOFT_HOME/lib:/usr
/GNUstep/Local/Library/Libraries:/usr/GNUstep/System/Library
/Libraries:/usr/java/jdk1.8.0_20/jre/lib/amd64/server' >> /home
/moogtoolrunner/.bashrc echo
$DONN
echo

echo $DIVD
echo "LOG: changing ownership of init scripts"
sed -i 's/PROCESS_OWNER=moogsoft/PROCESS_OWNER=moogtoolrunner/g' /etc
/init.d/*
echo "doublecheck: "
egrep PROCESS_OWNER= /etc/init.d/* --color echo $DONN
echo
```

2. Run the file on a clean install of Cisco Crosswork Situation Manager.

Tool Runner

In Cisco Crosswork Situation Manager the tool runner is used to run non-interactive command line tools that do not require root permission e.g. ping, cat. The output of running a tool will be asynchronously sent back to the web browser.

All commands are run in a Linux shell via ssh and executed on a specified user@hostuser@host.

When a tool is run from the UI, a user/password@host can be specified, but if not provided then the tool will be run on a default user@host (configured in [web.xml](#)).

Starting the tool runner through the Cisco Crosswork Situation Manager UI

From the Alert View:

- Right-click an Alert to open the menu
- Select **Tools** to open the tool runner

For each Alert (at present based on Alert type), a list of configured tools will be displayed with the following fields:

Field	Description
Tool	Display name of the tool
Run host	If left blank, the tool will run either on localhost, or, the default machine (which you can configure). If specified, the tool will run on the specified host
Username	If a run host is specified, then the username of the user that will be used to ssh into the run host needs to be specified
Password	If a username is specified, then this is the password that is used to log in to the run host. If left blank, the system will attempt to use ssh public/private keys
Fire & forget	If checked, then the tool will run and all output is ignored (basically the equivalent of 'nohup &')

Tool Runner Servlet Configuration

The tool runner is implemented as a servlet which can be configured using [web.xml](#) or [MySQL](#).

Supporting Configuration

Depending upon the exact environment, some support configuration may be required:

- The toolrunner uses ssh to login to a system with a certain username and password to run tools/integrations. As a result, 'PasswordAuthentication' must be set to 'yes' in /etc/ssh/sshd_config for this to work. Changes to that file require restarting the sshd service before they take effect.

MySQL

The database contains one table for the configuration of tools. The schema is as follows:

Element	Type	Description
tid	[INT AUTO_INCREMENT]	Alert Tool Id
displayname	[VARCHAR(100)]	Tool Display Name (the name displayed in the UI)
cmd	[VARCHAR(250)]	Tool Command (the actual command with no arguments)
args	[VARCHAR(500)]	Arguments to the command
alerttype	[TEXT]	Alert type of the alert that is used to determine the list of valid tools
description	[TEXT]	A textual description

When first installed, the table does not contain any tools. Tools can be added using the following SQL:

```
insert into moogdb.alert_tools value( 0,'Ping localhost', 'ping', '-c 5 localhost','SWBFence', 'This is an example tool');
```

Args

The args can be null if no argument is required. In addition, the argument can be configured to substitute information from the Alert, for example:

```
cmd = ping  
args = -c 5 $source -> -c 5 polyanna (after substitution)
```

You can use any Alert internal field name in the argument substitution e.g., \$source,\$source_id, \$severity (The case should match the internal field name).

Escaped arg variables

When an argument is substituted into the argument string it will also be escaped using a backslash. This allows you to substitute values to contain special characters, and also provides some security against command insertion. Be careful, the combination of the escaped value and the original argument string can produce unexpected results. For example, the command echo with the following argument strings will produce slightly different results:

```
argsV1 = $source argsV2  
= "$source"  
If $HOST = local.host resultV1=  
local.host resultV2 = local\host
```

Only the values will be escaped, not the original argument string. For further information on how to build the full argument string, read the next section.

Combining multiple argument into a single argument string

The following command has 2 arguments:

```
printf "%s\n" "hello world"
```

If the 'hello'and 'world'were from \$X and \$Y, the configuration for the command would be as follows:

```
cmd = printf  
  
args = "%s\n" $x\$y (version 1)
```

Alternatively:

```
args = "%s\n" "$x $y" (version 2)
```

Due to the argument value escaping, version 1 is probably the better choice (although version 2 is probably more readable/natural).

web.xml

Here is an example **web.xml** (/usr/share/apache-tomcat/webapps/toolrunner/WEB-INF):

```
<web-app>  
<servlet>
```

```
<servlet-name>toolrunner</servlet-name>
<servlet-class>com.moogsoft.toolrunner.CToolRunner</servlet-class>
<async-supported>true</async-supported>
<init-param>
<param-name>webhost</param-name>
<param-value>https://localhost</param-value>
</init-param>
<init-param>
<param-name>polluri</param-name>
<param-value>/poll</param-value>
</init-param>
<init-param>
<param-name>toolrunuri</param-name>
<param-value>/run</param-value>
</init-param>
<init-param>
<param-name>toolstopuri</param-name>
<param-value>/stop</param-value>
</init-param>
<init-param>
<param-name>sshtimeout</param-name>
<param-value>900000</param-value>
</init-param>
<init-param>
<param-name>toolrunnerhost</param-name>
<param-value>localhost</param-value>
</init-param>
<init-param>
<param-name>toolrunneruser</param-name>
<param-value>toolrunner</param-value>
</init-param>
<init-param>
<param-name>toolrunnerpassword</param-name>
<param-value>toolrunner</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>toolrunner</servlet-name>
<url-pattern>/poll</url-pattern>
<url-pattern>/run</url-pattern>
<url-pattern>/stop</url-pattern>
</servlet-mapping>
<listener>
<listener-class>com.moogsoft.toolrunner.CSessionListener</listener-class>
</listener>
</web-app>
```

The following parameters can be edited; however, these are 'global' settings and would apply to all tools:

webhost

Defines the value of the HTTP header 'Access-Control-Allow-Origin' that is used to control cross-origin resource sharing (CORS). The default value is <https://localhost>.

sshtimeout

If a tool running via ssh stops providing any output for a specified period of time, due to the tool hanging or a problem with the connection, sshtimeout can be used to shutdown any resources associated with the tool. This configuration value is specified in milliseconds and the default value is 900000 (1000 * 60 * 15 = 15 minutes).

toolrunnerhost / toolrunneruser / toolrunnerpassword

These three configurations relate to the running of tools 'locally'. From the UI, if only a tool name is specified then use/this configuration comes into play. There are basically two valid configurations:

- All three configured: Any tools with no run host specified will run on the configured run host. The system will ssh to the run host using the configured username and password
- Run host and username configured: If no password is configured, then the system will attempt to ssh to the run host using the configured username with a private key that must be located in a specific directory. For further information refer to Password-less Ssh

Nothing configured is no longer valid. This concerned any tools with no run host specified to run on the web server; however, now if nothing is configured, the tool runner will fail to start up.

SSH

ssh must be installed on the remote machine that you want to run tools on.

Enable Password Authentication

In order to successfully log in to a remote machine, the system requires that the remote ssh is configured to allow password authentication. In **/etc/sshd_config** configure the following:

```
>PasswordAuthentication yes
```

Passwordless Ssh

To setup a user account that does not require a password, you need to do the following:

1. On the web server machine generate a secure SSH key by typing: ssh-keygen.
2. Copy the generated key to the remote server. For each user account you plan to use to run tools you want to setup password-less logins with. Use the following command string, or, you can use scp: cat ~/.ssh/id_dsa.pub | ssh user@remotehost 'cat >> ~/.ssh/authorized_keys'. This command takes the generated SSH key from the local machine, connects to the remote host via SSH, and then uses cat to append the key file to the remote users authorized key list. As this connects with SSH to the remote machine, you will need to enter the password to use this command.
3. Confirm that you can now login to the remote SSH server without a password: ssh user@remotehost.com.
4. Copy the private key (`id_rsa`) to a directory called **keys**, which you need to create in the tomcat home directory, for example, `/export/apache-tomcat-7.0.29`.

Miscellaneous

- At present stdout and stderr are combined together and displayed in the web ui
- If a tool is run that produces a large amount of output, it can take a significant amount of time for the output to be processed over to the web ui so expect a time lag
- It is possible to run tools that require root permissions; however, this requires that a user is set up with root permissions or sudo permissions with no password. For example, in suds file => user ALL=(ALL) NOPASSWD: ALL

Custom Info

- Adding Custom_Info Fields
 - Adding Custom_Info Example
- Filling Custom_Info Fields
 - Filling Custom_Info Example
- Removing Custom_Info Fields
 - Removing Custom_Info Example
- Configure Custom_Info Search

Custom_info fields are customizable fields relating to either an Alert or a Situation that can be added to Cisco Crosswork Situation Manager during configuration. These will be displayed in the UI as columns in the Alerts and Situations Views and can be configured with optional sorting and filtering.

Please note: Custom_Info commands can be found in the usr/share/moogsoft/bin/utils folder

Adding Custom_Info Fields

The following commands can be used to add either Alert or Situation custom_info fields:

Command	Description
moog_add_alert_custom_field	This adds a new Alert custom_info field
moog_add_sitn_custom_field	This adds a new Situation custom_info field

To configure the display name, the field name and indexing, there are a number of options that can be used:

Option	Description
-d, --display_name <arg>	The display name of the field in the UI
-f, --field <arg>	The custom_info field name
-i, --index	This indicates the field is indexed for filtering and sorting
	<p>Please note: This cannot be used with display only fields</p>
	<p>If you are planning to use this custom_info field in Alert or Situation filters or you are planning to sort using this column we recommend you use the --index option to aid filter loading performance Too many indexed columns may affect the performance of additions</p>
-l, --loglevel <arg>	Specify (INFO WARN ALL) to select the amount of debug output
-o, --display_only	This indicates the field is for display only and cannot be used to filter, sort or search
-s, --size <arg>	The index size (the number of characters). This is valid for indexed text fields only. The default is 50
-t, --type <arg>	The type of field (number or text). The default is number

The example below shows how to add an alert custom_info text field which is also an indexed so will be filterable:

```
[root@moogsoft ~]# moog_add_alert_custom_field -d newfield -f new_field -i
-t TEXT
```

Adding Custom_Info Example

The addition of the new custom_info field will be confirmed with a message similar to the following:

Field newfield was added to UI successfully

Filterable field custom_info.new_field was added successfully

Filling Custom_Info Fields

There is a utility that allows you to fill the Alerts or Situations filterable custom_info fields using retrospective data:

Command	Description
moog_fill_alert_custom_field	This fills the filterable Alert custom_info fields using retrospective data
moog_fill_sitn_custom_field	This fills the filterable Situation custom_info fields using retrospective data

The amount of time the fill utility goes back and the log level can be configured using the following options:

Option	Description
-b, --back <arg>	This defines how far back the fill utility will go back, with 's' for seconds, 'm' for minutes, 'h' for hours, 'd' for days and 'w' for weeks E.g. -b 2w for two weeks
-l, --loglevel <arg>	<p>Please note: You can leave empty for all but this might take some time</p> Specify (INFO WARN ALL) to choose the amount of debug output

Filling Custom_Info Example

The example below shows how to fill Situation custom_info fields with retrospective data from the past three days:

```
[root@centos7 ~]# moog_fill_sitn_custom_fields -b 3d Filterable custom info
data was filled successfully
```

Removing Custom_Info Fields

The following commands can be used to remove previously configured Alert or Situation custom_info fields:

Command	Description
moog_remove_alert_custom_field	This removes a Alert custom_info field
moog_remove_sitn_custom_field	This removes a Situation custom_info field

After entering the command, type -f and enter the custom_info field name to select the field you want to remove.

Removing Custom_Info Example

The example below shows how to remove a custom_info field called 'new_field'.

```
[root@moogsoft ~]# moog_remove_alert_custom_field -f new_field Field
custom_info.new_field was removed successfully
```

Configure Custom_Info Search

You must run a utility if custom_info columns are added and existing Alerts or Situations contain values in that column for them to be filterable in the UI. Alert or Situations which are new or updated after the new column has been added will be filterable automatically.

If a Alert custom_info field has been added, run \$MOOGSOFT_HOME/bin/utils/moog_fill_alert_custom_fields

If a Situation custom_info field has been added, run \$MOOGSOFT_HOME/bin/utils/moog_fill_sitn_custom_fields

Data Archiving

Redirection Notice

This page will redirect to <https://docs.moogsoft.com/display/060500/Archive+Situations+and+Alerts>.

Data Parsing

- Parsing
 - Regex Parsing
 - Start & End Parsing
- Delimiters
- Variables
- Constants and conversions
- JSON events
- catchAll

Parsing

All received data is broken into tokens (tokenised) and then assembled into an Event. There are a number of parameters that allow you to control how tokenising works. The first two are a start and end character. The square brackets [] are the JSON notation for a list. You can have multiple start and end characters. The system considers an Event as all of the tokens between any start and end character.

```
start: [], end:  
["\n"],
```

The above example specifies:

- There is nothing defined in start; however, a carriage return (new line) is defined as the end character

In the example above, the LAM is expecting a entire line to be written followed by a return, and it will process the entire line as one Event.

Carefully set up, you can accept multi-line Events.

Regex Parsing

Regular expressions can be used to extract relevant data from the input data. Here's an example definition of how:

```
parsing:  
{  
    type: "regexp",  
    regexp:  
    {  
        # See https://docs.oracle.com/javase/8/docs/api/java/util/regex  
        /Pattern.html  
        # for accepted Regular Expression syntax. pattern :  
        "(?m)^START: (.*)$", capture_group: 1,
```

```

    tokeniser_type: "delimiters", delimiters:
    {
        ignoreQuotes: true, stripQuotes:
        true, ignores: "", delimiter:
        ["\"", "\r"]
    }
}

}

```

Start & End Parsing

Delimiters

Delimiters define how a line is split into tokens. For example, if you have a line of text data, it needs to be split up into a sequence of sub strings that are referenced by position from the start. So if you were processing a comma-separated file, where a comma separates each value, it would make sense to have the delimiter defined as a comma. Then the system would take all the text between start and end and break it up into tokens between the commas. The tokens could then be referenced by position number in the string starting from one, not zero.

For example if the input string was “the,cat,sat,on,the,mat” and comma was used as a separator, token 1 would be “the”, token 2 “cat” and so on.

Be aware, there are complications when you come to tokenisation and parsing. For example, if you say comma is the delimiter, and the token contains a comma, the consequence is that the token containing a comma to be split into two tokens. To avoid this it is recommended that you quote strings. You must then allow the system to know whether it should strip or ignore quotes, hence the stripQuotes and ignoreQuotes parameters.

```

ignoreQuotes: true,
stripQuotes: false, ignores: "",
delimiter: [",","\r"]

```

The above example specifies:

- If you have strings that are quoted between delimiters, ignoreQuotes set to true will look for delimiters inside the quote. For example, <delimiter>“hello “inside quote” goodbye”<delimiter> gives a token [hello inside quote goodbye]
- Setting stripQuotes to true removes start and end quotes from tokens. For example, “hello world” gives a token [hello world]
- ignores is a list of characters to ignore. Ignored characters are never included in tokens
- Delimiter is the list of valid delimiters used to split strings into tokens

Variables

For each event in the file, there is a positioned collection of tokens. Cisco Crosswork Situation Manager enables you to name these positions so if you have a large number of tokens in a line, of which you are interested in only five or six, instead of remembering it is token number 32, you can call token 32 something meaningful.

```

variables: [
    { name: "Identifier", position: 1 },
    { name: "Node", position: 4 },

```

```

    { name: "Serial", position: 3 },
    { name: "Manager", position: 6 },
    { name: "AlertGroup", position: 7 },
    { name: "Class", position: 8 },
    { name: "Agent", position: 9 },
    { name: "Severity", position: 5 },
    { name: "Summary", position: 10 },
  ],

```

The above example specifies:

- position 1 is assigned to Identifier; position 4 is assigned to node and so on Positions start at 1,
- and go up rather than array index style counting from 0

This is important because at the bottom of the file, **socket_lam.conf** there is a mapping object that configures how Cisco Crosswork Situation Manager assigns to the attributes of the Event that is sent to MooMs, values from the tokens that are parsed. For example, in mapping there is a value called rules, which is a list of assignments.

```

rules:
[
  {
    { name: "signature",rule: "$Node:$Serial" },
    { name: "source_id",rule: "$Node" },
    { name: "external_id",rule: "$Serial" },
    { name: "manager",rule: "$Manager" },
    { name: "source",rule: "$Node" },
    { name: "class",rule: "$Class" },
    { name: "agent",rule: "$LamInstanceName" },
    { name: "agent_location",rule: "$Node" },
    { name: "type",rule: "$AlertGroup" },
    { name: "severity",rule: "$Severity", conversion: "sevConverter" },
    { name: "description",rule: "$Summary" },
    { name: "first_occurred",rule: "$LastOccurrence" ,conversion:
      "stringToInt" },
    { name: "agent_time",rule: "$LastOccurrence",conversion: "stringToInt" }
  ]

```

In the example above, the first assignment name: "signature",rule: "\$Node:\$Serial" ("\$Node:\$Serial is a string with \$ syntax) means for signature take the tokens called Node and Serial and form a string with the value of Node followed by a colon followed by the value of Serial and call that signature in the Event that is sent to the Cisco Crosswork Situation Manager.

You define a number of these rules covering the base attributes of an Event. For reference, Cisco Crosswork Situation Manager expects a minimum set of attributes in an Event that are shown in this particular section.

Constants and conversions

There are a number of these rules, such as severity where there is a conversion defined. The following example looks up the value of severity and returns the mapped integer on the other side.

```

sevConverter:
{
    lookup: "severity", input:
    "STRING", output:
    "INTEGER"
},

severity:
{
    "CLEAR" : 0,
    "INDETERMINATE" : 1,
    "WARNING" : 2,
    "MINOR" : 3,
    "MAJOR" : 4,
    "CRITICAL" : 5,
    moog_lookup_default: 3
}

```

The above example specifies:

- sevConverter, which references a conversion definition in the conversions object
- The sevConverter uses a look up table lookup: "severity" to reference a table named severitydefined in the constants section

In this example the conversion takes as its input a string with a textual value of severity. From this it looks in the severity conversion table for a matching value and then returns the mapped value converted to an integer

moog_lookup_default can be used to specify a default value when an event is received which does not map to one of the values listed.

If moog_lookup_default setting is not used and an event is received which does not map to one of the other specifically listed values, the event will **NOT** be processed.

stringToIntis used in a time conversion, which forces the system to turn a string token into an integer value.

```

stringToInt:
{
    input: "STRING",
    output: "INTEGER"
}

```

The filter defines whether Moog uses a LamBot. If you comment out the presend value, no filter is applied to the events produced and all events are sent unchanged to the MooMs bus. If a LamBot is defined, for every event the system assembles using this configuration, the event is passed to the LamBot (via the presend function defined in the LamBot).

```

filter:
{
    presend: "SocketLam.js"
}

```

Therefore you must define a presendfunction in your JavaScript file.

The return value of the presendfunction will determine whether the event is sent on to the MooMs bus. The presendfunction can also define sub-streams that events are sent out on, so events can be sent to different farmd's.

You can provide an additional parameter to presend called modules that takes a JSON list. The JSON list is a list of optional JavaScript files that are loaded into the context of the LAMBot and executed; thus, you can share modules between LAMs. For example, you can write a generic Syslog processing module that is used in both the Socket LAM and the log file LAM. You therefore do not need to needlessly replicate code in the Moobot for each of the LAM's.

JSON events

The other capability of all LAMs is the native ability to consume JSON events. You must have a start and end carriage return as it is expecting a whole JSON object following the carriage return.

Under parsing you have:

```
end: ["\n"],
```

For the delimiter you have:

```
delimiter: ["\r"]
```

JSON is a sequence of attribute/value, and the attribute is used as a name. Under mapping, you must define the following attribute builtInMapper: "CJsonDecoder". It automatically populates, prior to the rules being run, all of the values contained in the JSON object.

For example if the JSON object to be parsed was:

```
{"Node" : "acmeSvr01","Severity":"Major"...}\n
```

The attributes available to the rules in the mapping section would be \$Node="acmeSvr01", \$Severity="Major" and so on.

catchAll

If you have an attribute that is never referenced in a rule, for example "enterprise trap number" which is never mapped into the attribute of an event, they are collected and placed as a JSON object in a variable called defined here and passed as part of the event.

Determine Probable Root Cause

- How does PRC work?
- How does Cisco
- Crosswork Situation Manager learn? PRC Column

Probable Root Cause (PRC) is a machine learning process in Cisco Crosswork Situation Manager that identifies which Alerts responsible for

causing a Situation. PRC looks for patterns in user supplied feedback. It does not use 'Root Cause Analysis' techniques.

With Probable Root Cause:

- You can immediately determine where to begin troubleshooting and diagnosis as soon as you open a Situation by looking at the Probable Root Cause Alerts.
- You can resolve Situations quickly by examining the Top 3 Probable Root Cause Alerts appear under Next Steps in a Situation Room.

For a brief introduction watch the video below:

How does PRC work?

You manually label Alerts as either a Root Cause Alert or a Symptom Alert, the Cisco Crosswork Situation Manager PRC Model uses this data to predict Situation root causes.

Subsequently, when Cisco Crosswork Situation Manager generates Situations, it labels an Alert or Alerts as having a Root Cause Estimate. A Root Cause Estimate is always assigned even if the data set is small. Generally, the more data Cisco Crosswork Situation Manager has the more accurate it is. However, that data needs to be consistent and the model is only as effective as the data it is supplied with. For example, two conflicting labels will confuse the model. If you do not know the status of an Alert do not label it. You do not have to label every Alert.

How does Cisco Crosswork Situation Manager learn?

Machine Learning uses features like Severity, Host, Description and Class and takes the values of those features for all labelled Alerts and uses a Neural Network to estimate the Root Cause for all the Alerts in a newly created Situation. It does this even if that Situation has not been seen before based on the model and labelled data.

See [PRC: Configure and Retrain](#) for more information on training your model.

PRC Column

This column (Situation, Alerts Tab) shows the Probable Root Cause Estimate as a percentage of the Alerts in that Situation and is useful as a prioritisation aid. For example, the higher the value an Alert has, the higher the probability that the Alert is the root cause of the Situation

As Alerts are added to a Situation, the Root Cause is recalculated (Situation, Alerts list) and therefore the PRC column may change. The more accurate and consistent data you feed your model the more accurate the estimate.

Probable Root Cause Configure and Retrain

- Getting Started
- Root Cause
- Reset all Feedback

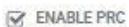
Getting Started

Go to **System Settings > Configure & Retrain**.

PRC is enabled by default. This means that you can mark Situation Alerts as having a root cause or not and Cisco Crosswork Situation Manager will show a root cause estimate in Next Steps in the Situation Room.

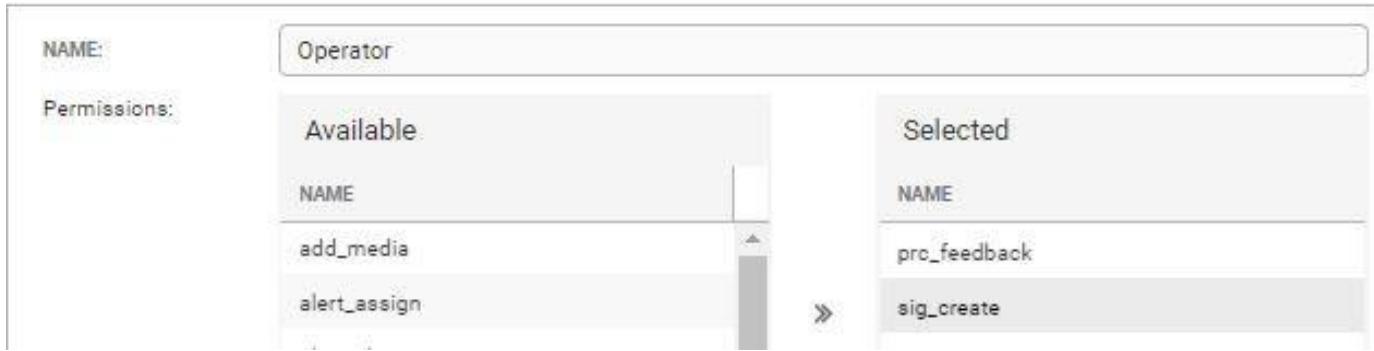
Probable Root Cause

Tick this option to enable Probable Root Cause



To configure which users can mark Alerts for PRC, go to **Security > Roles**. Select the role you want to edit and under Permissions, move 'prc_feedback' to 'Selected' using the direction arrows.

This permission is enabled for Administrator and Super User roles by default.



Root Cause

The PRC Model gives each Alert within a Situation a root cause estimate. Retrain recalculates the estimates with the current data.

Situation Root Cause

This Model gives estimates of how important an alert is within each situation.

RETRAIN

Reconfigure

AGENT (AGENT NAME / AGENT)

The agent of the alert represented as an enumeration. Each value of 'agent' is considered to be independent from all other values.

ALERT ARRIVAL ORDER (FIRST EVENT TIME / FIRST_EVENT_TIME)

Represents the arrival order of the alert in a situation.

ALERT TIME (FIRST EVENT TIME / FIRST_EVENT_TIME)

Represents the alert time as the components of the 'time of day' e.g. hours of day, minutes of hour.

CLASS (CLASS)

The class of the alert, represented in a way that identifies naming conventions in the class name.

DESCRIPTION (DESCRIPTION)

Tokenizes the description into words and uses those words to identify key words and phrases that may indicate root cause.

HOST (HOST / SOURCE)

The host of the alert, represented in a way that identifies naming conventions in the class name.

MANAGER (MANAGER)

The manager of the alert represented as an enumeration. Each value of 'manager' is considered to be independent from all other values.

SEVERITY & ARRIVAL ORDER (SEVERITY)

The severity of the alert represented as independent values and when the alert arrived for each value of severity. For best results use in conjunction with 'Severity Raw'.

SEVERITY ENUM (SEVERITY)

The severity of the alert represented as independent values. For best results use in conjunction with 'Severity Raw'.

SEVERITY RAW (SEVERITY)

The severity of the alert represented as a continuous value such that 'Warning' < 'Major' < 'Critical'. For best results use in conjunction with 'Severity Enum' or 'Severity & Arrival Order'.

SITUATION ALERT TIME (FIRST EVENT TIME / FIRST_EVENT_TIME)

Represents the alert time as the components of time e.g. hours of day, minutes of hour but relative the first event in the situation.

TYPE (TYPE)

The type of the alert, represented in a way that identifies naming conventions.

RECONFIGURE

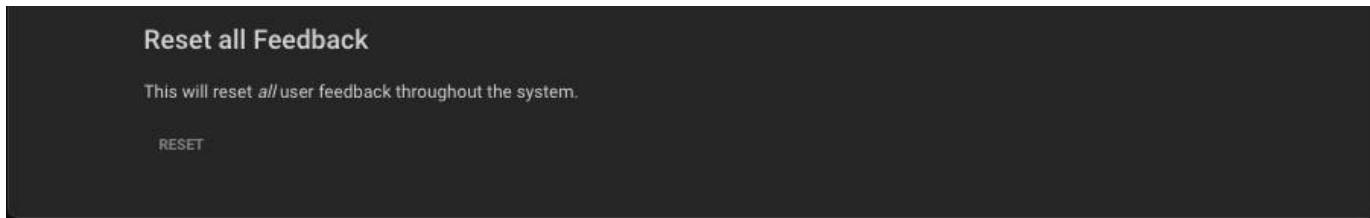
You can choose which features your model uses when predicting the root cause of an Alert. The default PRC configuration uses two types of Severity. The other features are listed below:

Feature	Description
Agent	The agent of the Alert represented as an enumeration. Each value of 'agent' is considered to be independent from all other values

Alert Arrival Order	Represents the arrival order of the Alert in a Situation
Alert Time	Represents the Alert time as the components of the 'time of day' e.g. hours of day, minutes of hour
Class	The class of the Alert, represented in a way that identifies naming conventions in the class name
Description	Tokenises the description into words and uses those words to identify key words and phrases that may indicate root cause
Host	The host of the Alert, represented in a way that identifies naming conventions in the class name
Manager	The manager of the Alert represented as an enumeration. Each value of 'manager' is considered to be independent from all other values
Severity & Arrival Order (default)	The severity of the Alert represented as independent values and when the alert arrived for each value of severity. For best results use in conjunction with 'Severity Raw'
Severity Enum	The severity of the alert represented as independent values. For best results use in conjunction with 'Severity Raw'
Severity Raw (default)	The severity of the alert represented as a continuous value such that 'Warning' < 'Major' < 'Critical'. For best results use in conjunction with 'Severity Enum' or 'Severity & Arrival Order'
Situation Alert Time	Represents the Alert time as the components of time e.g. hours of day, minutes of hour but relative the first event in the Situation
Type	The type of the Alert, represented in a way that identifies naming conventions

Reset all Feedback

To clear all feedback from your Cisco Crosswork Situation Manager instance click the **Reset**button:



Event and Alert Field best practice

This best practice is an attempt to offer consistency and reuse of configurations including the mapping from a source to an inbound event. The fields exposed in the alert/event are:

	Field	Required	Data Type	Size	Description	Example	Comment
1	signature	Yes	VARBINARY (binary)	767	This is a special attribute used to determine when Moog deduplicates events into Alerts. It can be any combination of one or more of the attributes listed below To be constructed as a subset of events from a source, also see existing guidance Constructed fields should be separated by ":" avoiding any	host1::nagios::cpu	

					possible issues with concatenation providing misleading results. e.g. NodeA event id 12 would concatenate as NodeA12, which would be the same as NodeA1 event 2, NodeA::12 and NodeA1::2 would therefore differentiate Signatures do not need to be human readable, so clarity isn't a concern. If length is becoming an issue - remove whitespace or other extraneous characters (via a lambot)		
2	alert_id	Yes	BIGINT(binary)	20	An auto-assigned incremental number. Internally generated DO NOT CHANGE		
3	source_id	Yes	TEXT(utf8)	65535	Source and Source_ID refer to the generating source of the event, primarily focused on the host environment. The Source should be any unique human readable name (FQDN, Hostname, etc) and the source_id should be any identifier for the source machine generated (IP, MAC, CI Number, etc.) If the event has no machine identification such as Application or other software generated events, then the Source should be some unique identifier of the instance (database name, cluster node, container name etc.). Again source_id should be any other unique identifier that is available (container UUID, cluster node UUID etc.) This attribute can be used for any additional identification attribute of the CI	192.168.1.107	
4	external_id	No	TEXT(utf8)	65535	Any unique identifier provided in the source event (event ID, Incident ID etc.) This is typically set to the CI's ID in the CMDB, or where the event is emitted from an underlying element management system, and may hold the unique source event identifier	12345	Returns Null if blank

5	manager	No	TEXT(utf8)	65535	A general identifier of the event generator or intermediary (NAGIOS, SCOM, etc.) In hub-and-spoke and/or relay architectures this typically is the name of the agent manager that pre-aggregates events prior to sending to Moog. For example, there may be an BMC Patrol manager that manages all San Francisco data center alerts. This field is also sometimes used simply to track the name of the Moog LAM that received the alerts in multi-LAM Moog deployments	Nagios	Returns Null if blank
6	source	Yes	TEXT(utf8)	65535	Source and Source ID refer to the generating source of the event, primarily focused on the host environment. The source should be any unique human readable name (FQDN, Hostname, etc) and the source_id should be any identifier for the source machine generated (IP, MAC, CI Number, etc.) If the event has no machine identification such as Application or other software generated events, then the Source should be some unique identifier of the instance (database name, cluster node, container name etc.). Again source_id should be any other unique identifier that is available (container UUID, cluster node UUID etc.)	host1	
7	class	Yes	TEXT(utf8)	65535	Class and Type are generic classifications for the event in a hierarchy that allow you to maintain a simple event ontologies; class then type. (Disk space: free space, Memory: max used..total available, etc.)	cpu	

8	agent	Yes	TEXT(utf8)	65535	The specific agent that created the event, (SCOM REST, NAGIOS SOCKET, SNMP TRAP NATIVE, etc.). This is typically the name of the agent that facilitates the event	Linux	
---	--------------	-----	------------	-------	---	-------	--

					from the CI e.g. "nagios-agent-london-7" A simple way to provide this is in the lam.conf by setting the agent: name and then mapping \$LamInstanceName to agent, <i>this is the default</i> { name: "agent", rule: "\$LamInstanceName" },		
9	agent_location	Yes	TEXT(utf8)	65535	This is typically the geographic location of the agent and/or CI such as "London". Should be used consistently for all sources, either as the host machine that the agent is executed from (BEM Server 1, OEM Monitor cluster, etc.) OR the physical location that the agent is executing (NYC Data Centre, Stuttgart Main Station, (51.407139, -0.307321) etc.)	New York, NY	

10	agent_time	Yes			This is the timestamp in epoch seconds when the event occurred. This should be set across all event sources to provide a common time reference. Timezones should be nullified - all events should be presented in the same time context. If an event source does not provide a suitable time in the payload then use the ingestion time at the LAM. Note: polled event sources (rest_client_lam, SCOM, Netcool) may skew the event time in line with the poll cycle. If an event is being generated in a different timezone and is manipulated into the Moog server time - add the origin time to the custom_info for the event. This can be operationally useful. e.g. custom_info.originalEventTime : agent_time should be in epoch seconds - convert as necessary. Miscalculated event times will cause unpredictable results across the system. Also see 4.1.2 Release note. [MOOG-2278] - Enhanced	
----	-------------------	-----	--	--	---	--

					Alert Times If the agent_time is not defined, it should be set to the current epoch time using Javascript functions such as: Math.round(Date.now() / 1000);		
11	type	Yes	TEXT(utf8)	65535	Class and Type are generic classifications for the event in a hierarchy that allow you to maintain a simple event ontologies; class then type. (Disk space: free space, Memory: max used..total available, etc.)	DOWN	
12	severity	Yes	INT(binary)	11	Standard 0-5 but be mindful of the significance across all event sources if possible. A low value event source could produce critical events that in the wider context would be considered minor Use the Cisco Crosswork Situation Manager LAM config file built in "sevMapper" to map your incoming severity values to a number between 0 and 5 : 0 = Clear 1 = Indeterminate 2 = Warning 3 = Minor 4 = Major 5 = Critical	5	0 clear - 5 critical
13	significance	No	INT(binary)	11	This value is calculated by moog Events Analyzer. Internally generated DO NOT CHANGE		
14	count	No	INT(binary)	11	The reference count of deduplicated Events for each Alert. Internally generated DO NOT CHANGE		
15	description	Yes	TEXT(utf8)	65535	The main text payload of the event. Add as much textual detail as possible. Remember a human operator will look at the detail and the entropy calculation works best with detailed narratives. Also see Creating Simple Canonicalized Description	CPU Threshold exceeded: 99%	

16	first_event_time	No	BIGINT(binary)	20	If you set agent_time in the LAM/LAMbot to the actual epoch seconds timestamp of each event, Moog will automatically keep track of the first and last occurrence of		
----	-------------------------	----	----------------	----	---	--	--

					multiple instances of the same event. Internally generated DO NOT CHANGE		
17	last_event_time	No	BIGINT(binary)	20			
18	int_last_event_time	No	BIGINT(binary)	20	Internally generated DO NOT CHANGE	1411134582	Fromagent_time
19	last_state_change	No	BIGINT(binary)	20	Internally generated DO NOT CHANGE		
20	state	No	INT(binary)	11	1 Opened 2 Unassigned 3 Assigned 4 Acknowledged 5 Unacknowledged 6 Active 7 Dormant 8 Resolved 9 Closed 10 SLA Exceeded Internally generated DO NOT CHANGE		
21	owner	No	INT(binary)	11	Set when an operator right- clicks on an alert in the Moog UI and assigns ownership. Internally generated DO NOT CHANGE		
22	entropy	No	DOUBLE(binary)	22	Internally generated DO NOT CHANGE		

23	custom_info	No	TEXT(utf8)	65535	<p>Custom_info is a special field that is the mechanism for extending the Moog alert schema. This is a JSON encoded string that should contain key value pairs for each data element not supplied in the initial event or having been enriched via alert transformation. Be consistent with key names so they can be used in Sigalisers and filters. Consider using a lambot module that sets a base set of custom_info across all lams - this provides a single point of administration for the customer. Care should be taken when setting custom_info in a LAM to ensure that it does not overwrite downstream additions (e.g. enrichment via a moobot) when the Event is de-duplicated.</p> <p>You can store simple or arbitrarily complex hierarchical JSON attributes in this field. They are basically serialized for use in the</p>	Returns Null if blank
----	--------------------	----	------------	-------	---	-----------------------

				standard JSON. parse/stringify manner and Moog's UI is written to display JSON hierarchies of any complexity in a tree- view format		
--	--	--	--	--	--	--

Event Processing

- **moog_farmd**
 - Command line attributes
 - Services
 - Configuration
 - system.conf
 - moog_farmd.conf
 - Moollets
 - Configuring threads

moog_farmd

moog_farmd is a core system application that is responsible for running all the algorithmics and automation relevant to the Cisco Crosswork Situation Manager product suite including:

- The creation of alerts
- The informatic analysis of those alerts to determine significance
- The clustering of alerts into Situations
- Any automation relating to the automated response (either), i.e., escalation, routing, notification, invitation of either alerts or Situations

You can run one or many moog_farmds on your system. (See [moog_farmd.conf](#) below.)

Command line attributes

The executable is a command line executable that can be run as a service daemon. The attributes can be viewed by typing:

```
% moog_farmd --help
```

Option	Description
--clear_state	Clears any persisted state information associated with this process group on startup
--cluster <arg>	Name of HA cluster (overwrites the value in the config file)
--group <arg>	Name of HA group (overwrites the value in the config file)
--leader <arg>	Is this instance an HA leader within its group (yes, no)
--mode <arg>	Start the process in passive or active mode (default will be active)
--config <arg>	Supplies a pathname to find the configuration file specific to the running instance
--instance <arg>	Enables you to give each instance of farmd a name, which you can use to refer to in farmd_ctrl. farmd_ctrl is a command line executable that allows you to bring up, restart and reload the various Moollets
--help	Displays all the command line options
--version	Displays the component's version number

--loglevel <arg>	Specifies the level of debug. By default, you get everything. In common with all executables in MOOG, having it set at that level can result in a lot of output (many messages per event message processed). In all production implementations it is recommended that log level be set to WARN, which only informs you of matters of importance
------------------	---

By default you do not need either 'config' or 'instance'. If you run the system without configuring either, the moog_farmd instance will load the default configuration file for moog_farmd, and respond to farmd_ctrl with no instance specified.

See [High Availability](#) for HA specific command line options.

Services

We advise that you start moog_farmd as a service. A service script is provided out of the box for the default moog_farmd configuration and is located here:

```
/etc/init.d/moogfarmd
```

A backup moogfarmd service script is located:

```
$MOOGSOFT_HOME/etc/service-wrappers/moogfarmd
```

If using multiple moog_farmds on the same host we advise you copy and modify the default moogfarmd service script for each moog_farmd running on the host.

1. Copy \$MOOGSOFT_HOME/etc/service-wrappers/moogfarmd to /etc/init.d/mymoogfarmd

Edit the following parameters in the /etc/init.d/mymoogfarmd file:

```
SERVICE_NAME=mymoogfarmd
CONFIG_FILE=$PROCESS_HOME/config/my_moog_farmd.conf
```

You now have a new service to be used to start your own specific moog_farmd.

```
service mymoogfarmd start
```

Configuration

system.conf

The general Cisco Crosswork Situation Manager system configuration file is located in **\$MOOGSOFT_HOME/config/system.conf**.

This file specifies how all components communicate with external systems such as MySQL and Elasticsearch. By editing the settings in this file, these systems can run on a separate host to that which runs the Cisco Crosswork Situation Manager software. The system is shipped with a default configuration, which assumes that MySQL and Elasticsearch are all running on the same host as Cisco Crosswork Situation Manager.

The format of the file is a JSON object, specifying the network locations of the various components:

```
...
"mysql" :
{
```

```

    "host" : "localhost", "database" :
    "moogdb", "username" : "ermintrude",
    "password" : "m00",
    "port" : 3306

},

"search" :
{
    "limit" "nodes" : 1000,
    [
        {
            "host" : "localhost",
            "port" : 9200
        }
    ]
}
...

```

moog_farmd.conf

If you are using many moog_farmds they each need their own config file. All instances of moog_farmd which do not specify a different **--config** use the default configuration file located in **\$MOOGSOFT_HOME/config/moog_farmd.conf**.

Moolets

moog_farmd runs **Moolets** which are individual isolated applications running inside a farmd app container. Moolets are a parallel concept to servlets in a traditional enterprise application container such as Tomcat. moog_farmd controls the flow of data through the Moolets where the data can come via the message bus or from other Moolets.

In the general configuration for moog_farmd there are certain parameters you need to configure, namely the number of threads available to each Moolet, followed by the list of each Moolet's specific configuration.

Configuring threads

The threads configuration element relates to the number of threads available to each Moolet. When you define a Moolet, if the system wide setting for threads is 10 (the default), farmd will run 10 instances of the Moolet so you can achieve high parallelism in the system. When you have an extremely high load, for example very high event rates (> 100 events per second), then you are advised to increase the number of threads. It may also be worth overriding the default value in specific Moolet configurations to achieve different levels of parallelism. E.g. You may give the Alert Builder Moolet more threads compared to the Sigaliser Moolet because you expect high levels of deduplication in the Alert Builder.

MooBot Modules

- Threads and global scope
- MooBot modules
- Worked examples
- Using external modules in MooBots
- onLoad function in MooBots

There are two places within Moog Cisco Crosswork Situation Manager where 'Bots' (simple computer programs for performing automated tasks) are used; in a moog_farmd Moolet, where they are called MooBots and in LAMs, where they are called LAMBots (For more information on LAMs and LAMBots, see [Data ingestion](#)). A MooBot is a JavaScript file that is loaded at start up by the Moolet. The MooBot exposes logic and data flow, which you can control in JavaScript, relevant to the necessary function.

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 automations.

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

Module	Description
Constants	Build a key value dictionary shared across Moobots.
ExternalDb	Access external relational databases.
Events	Set the types of Event that interest a Moobot.
Logger	Write log messages to the common moog_farmd log file.
MoogDb.V2	Query and manipulate a variety of entities in the Cisco Crosswork Situation Manager database, including alerts and Situations.
Process	Run and control the execution of other processes.
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](#).

Worked 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.jsclosure` instruct a MooBot to load into its Global Scope the available modules. For example, they can be used to:

- Change and create structure in the moogdb database
- Listen for specific Events in the system
- Push Events out
- Log to the common log file output
- Communicate using communication methodologies such as tweets, email etc.

Before you can use any of the built in modules that correspond to the functionality Cisco provides, you need the `preload()` method in the global object (`MooB.js`) to load the required modules.

The object exposes an API that you can use to add functionality into the system. In the example above, "Process" has a number of functions that you can call which allow the Moolet to run processes in the system.

After loading and running the `MooB.jsclosure` 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 in MooBots

MooBots can load external JavaScript modules. This means that modules can be reused as generic functions in multiple MooBots.

To do this:

- Add the external JavaScript module file (BotExampleModule.js) in the **\$MOOGSOFT_HOME/bots/moobots** or the **\$MOOGSOFT_HOME/contrib** directory
- Load the external JavaScript module in the MooBot by adding a line at the beginning (relative paths are supported), for example:

```
MooBot.loadModule('BotExampleModule.js');
```

The example below shows the external JavaScript module (BotExampleModule.js). It defines a class which takes an Alert and prints out a message:

```
function CPrinter()
{
    var mLogger=MooBot.loadModule('Logger'); var self=
    {
        prettyPrint: function(alert)
        {
            mLogger.info("This is a print of " + alert.value("alert_id") + " other info");
        }
    };
    var F=function() {};
    F.prototype=self; return( new
    F());
}
```

The AlertMgr.jsMooBot loads the external JavaScript module BotExampleModule.jsand 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 in MooBots

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: '102.168.1.141'
```

```

        port: '3306', database:
        'emp_db'
    },
    customers: {
        type: 'sqlServer', host:
        '213.32.112.17',
        database: 'customers', user: 'sa',
        encrypted_password:
        '0rJGl5oCwpmE9Hbk32sxFgxIQV3O5cx2bx1vKNOM7YA='
    }
};

}

```

Config Bot Module

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.

Prerequisites

Before you use the Config Bot module, ensure the following conditions are met:

- The configuration file is in valid JSON.
- The configuration file is in \$MOOGSOFT_HOME/config
- The configuration is present on the file system as the process running the bot.

Config Bot Examples

If you wanted to create a URL that links to Cisco Crosswork Situation Manager Situations, the Config Bot module can be used to dynamically retrieve the base URL of the Cisco Crosswork Situation Manager instance from servlets.conf:

```

var config = MooBot.loadModule('Config');
.....
var servletsConf = config.getConfig('servlets.conf'); if (servletsConf) {
    moogURL = servletsConf.webhost;
}

```

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.|+

```

Best Practices

You should follow these guidelines when using the Config Bot module:

- It should only be used within the constraints of the OnLoad() function.
- Making multiple calls to this module may impact the performance of the Bot.
- Custom config files should be kept in a subdirectory of config and named appropriately.
- Comment any config file extensively so other users can understand the context of their use.

Constants

Description

Each MooBot runs in its own thread, and instances of MooBots are independent of each other. The Constants module enables the sharing of logic, states or flags between MooBots. It allows you to 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 you are interested in listening for. See the Event types table under **constants.eventType()**.

The Constants module is available to load into any standard MooBot.

To use, at the top of a MooBot js file, define a new global object constants to load the Constants module:

```
var constants = MooBot.loadModule('Constants');
```

Reference Guide

 [constants.put](#)

constants.put()

Associates the specified value with the specified key.

Previous key mapping has the old value replaced.

Request Argument

Name	Type	Description
key	String	The key with which the specified value is to be associated
value	Object	The value associated with the key

Return Parameter

Void - no value returned

 [constants.get](#)

constants.get(key)

Returns the value to which the specified key is mapped.

Request Argument

Name	Type	Description
key	String	The key indicating where to retrieve the value

Return Parameter

Type	Description

Object The value to which key is mapped, or nullif there is no mapping

↳ [constants.contains](#)

constants.contains()

Returns true if the module contains an object with that name.

Request Argument

Name	Type	Description
name	String	The object name

Return Parameter

Type	Description
Boolean	true if the module contains an object with that name, otherwise false

↳ [constants.remove](#)

constants.remove()

Removes the value to which the specified key is mapped

Request Argument

Name	Type	Description
key	String	The key with which the value will be removed

Return Parameter

Void - no value returned

↳ [constants.eventType](#)

constants.eventType()

Returns a CEvent object for a range of valid Event types.

Request Argument

Name	Type	Description
<name>	String	The name of the Event type. See below

Event types:

Name	Passed	Description
E_LamEvent	("Event"/"Events")	A raw Event from a LAM
E_NewAlert	("Alert"/"Alerts")	New Alert
E_AlertUpdate	("AlertUpdate")	Alert update
E_CloseAlert	("AlertClose")	Close Alert

E_NewComment	("Comment"/"Comments")	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")	The user name
E_Unknown	("Unknown")	An Event which is uncategorised (an error condition)

As Cisco enhances and develops the system, more Event types may be added.

Return Parameter

Type	Description
CEvent	A CEvent object containing the value for the specified Event type

Examples

The following example in **AlertBuilder.js**, shows the Constants module with two methods that allow you to post and retrieve values from a shared scratchpad.

```
var count=0 constants.put("counter",count);
```

The variable count is set to 0 and stored using the label counter.

- put ()takes the name of the variable you want to store counter, and the variable count.

You can later retrieve a value by calling the method get ()and passing the name of the shared attribute you want, which is returned as a JavaScript local variable.

```
var count_val=constants.get("counter"); count_val++;
constants.put("counter",count_val);
```

- get takes the name of the shared attribute, counter
- count_val is incremented

- `put` takes the name of the variable you want to store, `counter`, and the incremented value, `count_val`

If you do not have anything stored under `counter`, the MooBot returns the JavaScript object `null`.

eventType Example

The following `eventType()` example passes the name of an Event and returns a system wide Constant which identifies that type of Event when using the Events module.

```
constants.eventType("Event")
```

Events

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 Moo Bot.

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](#)

`events.onEvent()`

Takes the name of a valid JavaScript function in a MooBot and also an Event code (from the Constants module `eventType`), and returns an Event adaptor object

Request Arguments

Name	Type	Description
<code>functionName</code>	String	The name of a valid JavaScript function in the MooBot that is called when the Event arrives
<code>type</code>	CEvent	<code>eventType</code> 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 <code>listen()</code> function in-line to listen for the Event type

Example

For the AlertBuilder MooBot:

```
events.onEvent("newEvent",constants.eventType("Event")).listen();
```

- Call the `newEvent` Javascript function
- Define the Event type `Event` (from the [Constants module](#)), which responds to Events put on the MooMS bus by a LAM
- 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:

↳ `function newEvent`

function newEvent()

Request Argument

Name	Type	Description
event	CEvent object	An object that encapsulates all the data for the Event from the bus, and allows you to forward the Event to the bus, using the CEvent forward method detailed below

CEvent forward methods

You can emulate MoogDb behaviour by running the MoogDb.V2 MooBots.

For example, the `alert.forward(this)` line will send an Alert onto the Moolets specified in the appropriate `process_output_of` block within `moog_farmd.conf`

You could instead remove the `process_output_of`s from the AlertRulesEngine / Sigaliser / Cookbook / Speedbird / Nexus Moolets and explicitly send Events / Alerts / Situations on within the MooBot code using (as an example)

```
alert.forward("Cookbook");
```

The advantage of this approach is that Alerts / Situations can be forwarded to different AlertRulesEngines / Sigalisers dynamically in the MooBots (for example based on the value of the source field)

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 MooMS 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](#)

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](#)

set()

Associates the specified value with the specified name in the Event.

Previous keymapping 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](#)

value()

Returns the object stored at the key name.

Request Argument

Name	Type	Description

name	String	The name of the key to return the object from
------	--------	---

Return Parameter

Type	Description
Object	A Javascript object containing what is at the key name

Note

Compatibility with MoogDb and MoogDb.V2

Methods and auxiliary objects listed here are compatible with the [MoogDb.V2](#) module

CEvents API

Cisco Crosswork Situation Manager uses the CEvents API to pass data

to LAMbotfunctions. The methods are:

`type`

`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

Name	Type	Description
N/A	N/A	N/A

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`

`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.

`payload`

`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

Name	Type	Description
N/A	N/A	N/A

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}
```

evaluateFilter

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'");
```

`stringValue`

`stringValue()`

Fetch a value from inside the payload which matches the provided key as a string value.

Request Arguments

Name	Type	Description
name	String	The key for a value stored in the payload which will be used to fetch the data.

Return Parameter

Type	Description
String	The value from the payload that was stored alongside the key (or null if no value was found to for the provided key) which has been converted to string format.

`contains`

`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.

`set`

`set()`

Used to insert or update a value in the payload of the CEvent.

Request Arguments

Name	Type	Description
nm	String	The key to insert or change a value at.
val	Object	The new value to store against the key.

Return Parameter

Type	Description
boolean	Whether or not the value was successfully changed.

`getJournalDetails`

`getActionDetails()`

A utility helper method provided to retrieve the entire alert or Situation contained in the payload of a CEvent. Predominantly intended for the Journalling Moobot.

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.

[getCustomInfo](#)

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.

[setCustomInfo](#)

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](#)

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
Boolean	Indicates if the operation was successful: true = success, false = fail

[getCorrelationInfo](#)

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.

[getPreviousData](#)

getPreviousData()

Fetches the previous attribute data for a modified alert or Situation.

Request Arguments

None.

Return Parameter

Type	Description
JS NativeObject	The attribute data that existed immediately prior to the alert or Situation change. The timestamp last_state_change indicates when the change was made.

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](#)

getSummaryData()

Fetches a summary of information about a system, such as the number of alerts or the service count bundled up as key/value pairs.

Request Arguments

Name	Type	Description
replacementValue	String/integer/boolean/object/map	The string or string/integer/boolean/object/map value to replace the value stored in the custom_info field.

Return Parameter

Type	Description
JS NativeObject	The summary of information about a system: summary.alert_count - number summary.service_count - number summary.sig_summaries - map (contains "categories" and "queues") summary.sig_summaries.categories - (array of objects) summary.sig_summaries.queues - (array of objects) categories and queues contain the following: sig_total - number, alert_total - number, name -string summary.sigs_down - number summary.sigs_up - number summary.total_events - number summary.total_sigs - number

↳ [getTopic](#)

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".

↳ [setTopic](#)

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

↳ [forward\(moobot\)](#)

forward(moobot)

This will forward the CEvent down the chain configured in the moog_farmd.conf (using process_output_of configuration), usual way of calling this is CEvent.forward(this) where this is the Moobot that's processing the CEvent object.

Request Arguments

--	--	--

Name	Type	Description
moobot	NativeObject	The instance of the Moobot which is handling the CEvent object.

Return Parameter

None

↳ [forward\(target,...\)](#)

forward(target,...)

Takes any number of target moolts names as strings and forwards the CEvent to each of them. For example CEvent.forward("moolt1") or CEvent.forward("moolt1", "moolt2").

Request Arguments

Name	Type	Description
targets	String Varargs	Any number of String Moolt names which the CEvent is going to be forwarded to.

Return Parameter

None

Common fields for both Situations and alerts in the payload:

- custom_info
- description
- first_event_time
- last_event_time
- last_state_change

Events (MOOGDb only)

Compatibility with MoogDb and MoogDb.V2

Methods and auxiliary objects listed here are compatible with the [MoogDb](#) module, which was removed in v4.1.14
Information here is provided for reference only

Those methods and auxiliary objects compatible with its replacement 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](#)

events.onEvent()

Takes the name of a valid JavaScript function in a MooBot and also an 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	<code>eventType</code> 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 <code>listen()</code> function in-line to listen for the Event type

Example

For the AlertBuilder MooBot:

```
events.onEvent("newEvent",constants.eventType("Event")).listen();
```

- Call the `newEvent` Javascript function.
- Define the Event type `Event` (from the [Constants module](#)), which responds to Events put on the MooMs bus by a LAM.
- 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](#)

```
function newEvent()
```

Request Arguments

Name	Type	Description
event	CEvent object	An object that encapsulates all the data for the Event from the 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 MooMs 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 MooMs 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

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

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`

`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`

`getJournalDetails()`

Returns the details (if any) of the journaled operation for a Situation

Request Argument

Name	Type	Description
scope	Javascript object	The MooBot context, provided by using this as a parameter

Return Parameter

Type	Description
Object	Javascript object containing the details of the journaled operation for a Situation

`getCustomInfo`

`getCustomInfo()`

Returns the custom information (if any) for an Alert or Situation.

Request Argument

Name	Type	Description
scope	Javascript object	The MooBot context, provided by using this as a parameter

Return Parameter

Type	Description
Object	Javascript object containing the custom information

[setCustomInfo](#)

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...](#)

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...](#)

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 is being deprecated

 [message](#)

message()

Object to broadcast on

Request Argument

Name	Type	Description
msg	CEvent	Object to broadcast on

Return Parameter

Void - no value returned

 [topic](#)

topic()

Topic to broadcast message on

Request Argument

Name	Type	Description
topic	String	The topic name

Return Parameter

Void - no value returned

 [output](#)

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](#)

retcode()

The retcodevalue 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

doNotPropagate()

Indicates that no propagation is needed

Request Argument

None.

Return Parameter

Void - no value returned

ExternalDb

- **Description**
 - Methods
- **Reference Guide**
 - externalDb.connect()
 - externalDb.execute()
 - externalDb.query()
 - externalDb.prepare()
- **Database specific information**
 - Important Notes on downloading JDBC Drivers
 - Microsoft SQL Server:
 - MySQL:
 - IBM DB2:
 - Oracle:
 - 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):

- MySQL
- Microsoft SQL Server
- IBM DB2
- Oracle
- PostgreSQL

With ExternalDb, Cisco Crosswork Situation Manager can retrieve information from external databases for use in Alerts and Situations and can also update information in external databases with information from Cisco Crosswork Situation Manager.

ExternalDb is available to load into any standard MooBot.

To use, at the top of a MooBot.js file, define a new global object externalDb to load the ExternalDb module:

```
var externalDb = MooBot.loadModule('ExternalDb');
```

externalDb is the name of the database.

Methods

- externalDb.connect
- externalDb.execute
- externalDb.query
- 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 connectmethod 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, loginTimeoutfor SQL Server or useCompressionfor 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 MOOG Cisco Crosswork Situation Manager install.
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 the external database by including the pool_size key
pool_size	The number of connections in the pool. Must be 1 or more, defaults to 10. Generally, this should match the number of threads configured to run the MooBot

Note: You can also define connection properties in the configuration file moog_externalDbDetails.conf

Return Parameter

Type	Description
Object	A Java object containing connection details, depending on the requested connection properties Returns null if no connection is available (due to either misconfiguration or unavailability of the external database)

Notes:

The connectmethod can accept a single parameter with connection properties, or two parameters - one with the generic connection properties and one specific for this connection. For example:

```
var customersConnection = externalDb.connect(dbTypes.customers);
```

will connect to the customer database as is.

```
var customersConnection = externalDb.connect(dbTypes.customers, {user: 'admin', password: 'wrdPass'});
```

will connect to the same database as the 'admin' user, with the password supplied.

You can also use the name from the configuration file moog_externalDbDetails.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.

[^ Back to Methods](#)

externalDb.execute()

Performs an SQL update to the database.

The executemethod 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 querymethod has one string argument:

Request Argument

Name	Type	Description
argument	String	SQL string argument

Return Parameter

Type	Description
Object	A Java object which can containing the sub methods listed below Returns nullif 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 nullif 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");
}
```

[^ Back to Methods](#)

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 preparemethod 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()	<p>Close the prepared statement</p> <p>Note: It is important to close the statement with this method when no longer needed</p>

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();
```

[^ Back to Methods](#)

Database specific information

Important Notes on downloading JDBC Drivers

1. Be sure to download the correct version of the JDBC Driver for your database.
2. Ensure downloaded JDBC drivers are moved/copied to the \$MOOGSOFT_HOME/lib/cots directory.

Microsoft SQL Server:

JDBC driver: <http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774>

Connection properties: [http://technet.microsoft.com/en-us/library/ms378672\(v=sql.110\).aspx](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/tjvjccn.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: "moogsoft" }  
    }
```

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

Introduction

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_builderutility, 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:

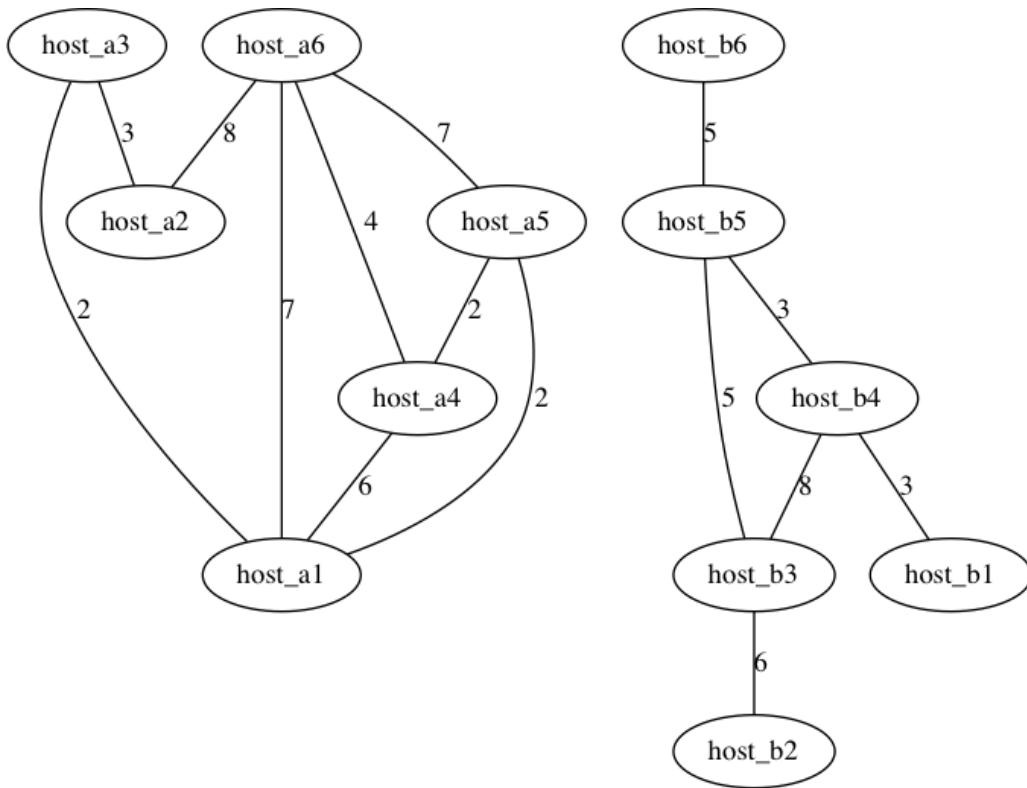
```

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

```

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

[topo.loadTopology](#)

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

Running:

```
var ret = topo.loadTopology();
logger.warning("loadTopology -> " + ret);
```

...returns the output below. The topology loaded successfully:

```
WARN : ... [CLogModule.java]:99 +|loadTopology -> true|+
```

 [topo.isConnected](#)

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](#)

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](#)

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: <ul style="list-style-type: none">• 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)

The Distance (shortest path) between the nodes host_a5 and host_a6 is **6**, and the output below is returned:

```
WARN : ... [CLogModule.java]:99 +|distance 1 -> 6|+
```

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_b2 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|+
```

If the radius had been specified as 16 or above, a Distance result (not -1) would be returned; because the Distance (shortest path) between nodes host_b2 and host_b6 is 16.

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](#)

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 connections

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](#)

addEdge(String sourceNode, String sinkNode)

Optional parameter: Double weight (default value=1.0)

These will add a new node to a topology/graph both in memory and in the database.

Behaviour:

- if unspecified, weight will have default value 1.0
- any new nodes will be saved in memory and db
- new connection will be saved in memory and db

GraphTopo:

- won't work if there already is such edge
- uses jgraph methods addVertex and addEdge

Topo:

- won't work if both nodes aren't in topology or if both nodes already are in
- does not recalculate a topology, new coordinate == old coordinate + weight
- new coordinates will be saved in memory and database

Logger

The Logger module sets the log level in moog_farmd, allowing log messages to be written to the common moog_farmd log file.

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 (for more information click [here](#)) have been deprecated in favour of the 'single string argument' version

↳ [logger.debug](#)

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](#)

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](#)

logger.warning()

Sends a warning log message (a higher severity level). For example, this can be used to log behaviour 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](#)

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 moog_farmd 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.

It can be used load 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 in the sections below.

Methods

- `mailer.initTransport(mailObj)`
- `mailer.send(mailMsg)`

mailer.initTransport(mailObj)

Defines the mail server information needed to send the email in the send function.

Request Argument

Name	Type	Description
mailObj	Object	A JSON object specifying connection properties

Example

```
mailer.initTransport({  
    server: "smtp.mailserver.com", port: 2525,  
    account: "user@mailserver.com",  
  
    password: "m00gsoft",  
    // Set to true, to decrypt the above password value.  
    // Use the moog_encryptor utility to encrypt a password or secret value.  
    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)

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

```
//  
// Ok, we must now construct a message to be sent by the mailer  
  
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  
  
};  
  
//  
// Send it  
// mailer.send(mailMsg);
```

MoogDb V2

You can query and manipulate a variety of entities in the Cisco Crosswork Situation Manager database using the MoogDb.V2 MooBot module.

The module uses various methods to retrieve information from Moogdb and update components of Cisco Crosswork Situation Manager including alerts, Situations, users and teams.

All MoogDb.V2 methods that update the database also publish information about the appropriate updated entities on the [Message Bus](#), so any updated information automatically appears in Cisco Crosswork Situation Manager when the relevant method is called.

Load MoogDB.V2

You can load the MoogDb.V2 module into any standard MooBot by defining a new global object called moogdb at the top of the JavaScript file:

```
var moogdb = MooBot.loadModule('MoogDb.V2');
```

Methods

All available MoogDb.V2 methods are described in the sections below:

[addAlertToSituation](#)

addAlertToSituation()

Adds a specified Alert to a Situation.

Request Arguments

Name	Type	Description
alertId	Number	The Alert ID
situationId	Number	The Situation ID

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

[addCorrelationInfo](#)

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](#)

addSigCorrelationInfo()

Adds correlation information (external service name and external entity ID) to a Situation. This is the recommended method for adding correlation information to a Situation, the addCorrelationInfo method has been retained for backwards compatibility.

Request Arguments

Name	Type	Description
situationId	Number	The Situation ID
service	String	The name of the external service, such as ServiceNow
externalId	String	The identifier that the entity has in the external service, which corresponds to the Situation

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

[addProcess](#)

addProcess()

Adds a new process to the database.

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

Request Arguments

Name	Type	Description
process	String	The process name
description	String	The process description.

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

[addService](#)

addService()

Adds a new external service to the database.

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

Request Arguments

Name	Type	Description

service	String	The name of the external service being added
description	String	The service description

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

↳ [assignAlert](#)

assignAlert()

Assigns an Alert to a valid user, identified by their user ID.

Request Arguments

Name	Type	Description
alertId	Number	The Alert ID
userId	Number	A valid user ID
username	String	A valid user name

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

↳ [assignAndAcknowledgeAlert](#)

assignAndAcknowledgeAlert()

Assigns an Alert to a valid user, identified by their user ID, and acknowledge the alert.

Request Arguments

Name	Type	Description
alertId	Number	The Alert ID
userId	Number	A valid user ID
username	String	A valid user name

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

↳ [assignAndAcknowledgeSituation](#)

assignAndAcknowledgeSituation()

Assigns a Situation to a valid user, identified by their user ID, and acknowledge the situation.

Request Arguments

Name	Type	Description
situationId	Number	The Situation ID
userId	Integer	A valid user ID
username	String	A valid user name

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

↳ [assignModerator](#)

assignModerator()

Assigns a Situation to a valid user, identified by their user ID.

Request Arguments

Name	Type	Description
situationId	Number	The Situation ID
moderatorId	Number	A valid user ID
username	String	A valid user name

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

↳ [assignTeamsToSituation](#)

assignTeamsToSituation()

Assigns one or more teams to a Situation. Once successfully run, Cisco Crosswork Situation Manager marks the Situation as overridden and the Teams Manager Moolet can no longer modify its team assignment. See [Teams Manager Moolet](#) for more information.

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

Request Arguments

Include either team_ids or team_names.

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

Return Parameter

Type	Description

Native object

A Javascript object containing a list of the team names or team IDs assigned to the Situation, depending on the input.

Input Example 1:

```
var assignTeamIDs = moogdb.assignTeamsToSituation(1, { "team_ids" : [1, 2] } )
```

Return:

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

Input Example 2:

```
var assignTeamNames = moogdb.assignTeamsToSituation(2, { "team_names" : [ "Team1", "Team2" ] } )
```

Return:

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

Unassign Example:

```
var unassignTeamIDs = moogdb.assignTeamsToSituation(1, { "team_ids" : [] } )
```

Return:

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

↳ [closeAlert](#)

closeAlert()

Closes one or more alerts.

Request Argument

Name	Type	Description
alertId	Number	A single alert ID
alertIds	List	A list of alert IDs
thread_entry_comment	String	Optional comment

Return Parameter

Type	Description

Boolean	Indicates if the operation was successful: true = success, false = fail
---------	---

Example Input

```
var success = moogdb.closeAlert([78,234,737], "Closing as agreed during team discussion 1/1/2018");
```

↳ [closeSituation](#)

closeSituation()

Closes a Situation.

Request Arguments

Name	Type	Description
situationId	Number	The Situation ID
closeAlerts	Constant	<p>Determines how the Alerts in the Situation are treated:</p> <ul style="list-style-type: none"> CLOSE_NO_ALERT- No Alerts are closed CLOSE_ALL_ALERTS- All Alerts are closed CLOSE_UNUSED_ALERTS- Only the Alerts unique to this Situation (i.e. otherwise unused) are closed <p>To access these constants from a MooBot, precede them with the module name, for example:</p> <p>moogdb.CLOSE_NO_ALERT</p>

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

↳ [createAlert](#)

createAlert()

Creates or updates an Alert in the database. Optionally updates custom info for de-duplicated Alerts.

Request Arguments

Name	Type	Description
alert	Native object	A Javascript object containing Alert attributes, such as type, severity, etc
event	CEvent	A CEvent object representing the Alert, containing Alert attributes, such as type, severity, etc
mergeCustomInfo	Boolean	Set this to 'true' to merge the custom_info data in this Alert with the info held in the database Optional

Return Parameter

Type	Description
CEvent	A CEvent object containing the latest version of the Alert

↳ [createMaintenanceWindow](#)

createMaintenanceWindow()

Creates a maintenance window that filters alerts, by passing an object containing the information.

Request Arguments

Name	Type	Description
maintenanceWindowObj	Object	A map containing the following information
name	String	Mandatory - The name of the maintenance window
description	String	Mandatory - The description of the maintenance window
filter	String	Mandatory - The filter to apply to the new alerts created
start_date_time	Number (Epoch)	Mandatory - The time in epoch where the maintenance window will start, up to a maximum of 5 years in the future.
duration	Number (seconds)	Mandatory -The duration in seconds where the maintenance window is running, must be greater than zero.
forward_alerts	boolean	Mandatory -whether the alert will be forwarded to situation or not
recurring_period	Number	Optional - How many days/weeks/months to wait before this recurs. The recurring_period property must be 1 - no other value will be accepted
recurring_period_unit	Number	Optional - Decides what the recurring period counts in 0 = minutes, 1 = hours, 2 = days, 3 = weeks, 4 = months. The recurring_period_units property has allowed values of 2 (daily), 3 (weekly) or 4 (monthly) - no other values will be accepted

Input example :

```
{
    "name": "Mike",
    "description": "A description",
    "filter": "{'column': 'source', 'op': 0, 'value': '\'Nile\'', 'type': 'LEAF'}",
    "start_date_time": 1497971059,
    "duration": 360000,
    "forward_alerts": true,
    "recurring_period": 1,
    "recurring_period_unit": 2
}
```

Return Parameter

Type	Description
Number	The window ID created, or null if an error occurred

↳ [createSituation](#)

createSituation()

Creates a new Situation, containing no Alerts.

Situation settings, such as severity are defined in the following arguments:

Request Arguments

Name	Type	Description
moderator	String	A valid user name
label	String	The new Situation description

Return Parameter

Type	Description
CEvent	The newly created Situation wrapped in a CEvent object

↳ [createTeam](#)

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
Number	The team id created, or null if an error occurred

createThread

createThread()

Creates a new thread for a Situation.

Threads are comments or 'story activity' on Situations (More information [here](#)).

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

createThreadEntry()

Creates an entry on the specified thread.

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](#)

createUser()

Create a user, by passing an object containing user properties

Request Arguments

Name	Type	Description
userObj	Object	A map containing the following user information
username	String	Mandatory - the new user (unique) login username
password	String	The new user password (only valid for DB realm)
active	Boolean	true if the user active, false if the user inactive, default to true
email	String	The user email address
fullname	String	The user full name
roles	JSON list	Mandatory - List of user roles. That list should contain either the list the role IDs or the role names. E.g "roles":["Super User"],
primary_group	String or Number	The user primary group name or primary group id
department	String or number	The user department id or name
joined	Number	The time the user joined (in Unix time)
timezone	String	The user timezone
contact_num	String	The user phone number
session_expiry	Number	The number of minutes after which the user session will expire. Default to system default
competencies	JSON list	A list with the user competencies. Each competency should have have name or cid and ranking. That is, something like:
		<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
--	--

Example

Input:

```
{
    "username": "user1",
    "fullname": "firstName surName",
    "competencies": [
        {
            "name": "SunOS",
            "ranking": 40
        },
        {
            "name": "SAP",
            "ranking": 50
        },
        {
            "name": "EMC",
            "ranking": 60
        }
    ],
    "roles": ["Super User"],
    "department": 3, "active": true,
    "email": "user@email.com",
    "timezone": "a timezone", "teams": [1,
    2, 4],
    "joined": 12345678,
    "contact_num": "0965412345"
}
```

Return Parameter

Type	Description
Number	The user id created, or null if an error occurred

[deAssignAlert](#)

deAssignAlert()

Deassigns an alert. The user assigned to the alert is removed, and the user is set to ANON.

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](#)

deleteMaintenanceWindow()

Delete an existing 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

[deleteMaintenanceWindows](#)

deleteMaintenanceWindows()

Delete existing maintenance windows.

Request Argument

Name	Type	Description
filter	String	The filter to delete windows by. Something like: description matches 'dfgvhbjk'
limit	Number	The maximal number of windows to fetch. default to 100.

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

[getActiveSituationIds](#)

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](#)

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

↳ [getAlertIds](#)

getAlertIds()

Get all alertids 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 and the alert IDs

Example

Return:

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

↳ [getSigCorrelationInfo](#)

getSigCorrelationInfo

Retrieves all correlation information related to a specified Situation.

Request Arguments

Name	Type	Description
sitn_id	Number	The Situation ID.

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

↳ [getMaintenanceWindows](#)

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 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  
    }  
  ]  
}
```

↳ [findMaintenanceWindows](#)

findMaintenanceWindows()

Find maintenance windows based on a filter and how many should be fetched.

Request Argument

Name	Type	Description
filter	String	The filter to find windows by. Something like: description matches 'dfgvhbjk'
limit	Number	The maximal number of windows to fetch. default to 100.

Return Parameter

Type	Description
NativeObject	A Javascript object containing the windows

Example

Return:

```
{  
  "windows": [  
    {  
      "filter": "{\"op\":6,\"column\":\"severity\",\"type\":\"LEAF\",\"value\":[2]}",  
      "duration": 3600,  
      "recurring_period": 1, "del_flag": false,  
      "forward_alerts": false, "last_updated":  
        1491917013, "name": "window1",  
      "updated_by": 3, "description":  
        "dfgvhbjk", "id": 1,  
      "recurring_period_units": 2,  
      "start_date_time": 1491916979  
    }  
  ]  
}
```

↳ [getQueueName](#)

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	The queue name

↳ [getPrcLabels](#)

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](#)

getProcesses()

Fetches a list of processes from the database.

Request Argument

Name	Type	Description
limit	Integer	The maximum number of processes to retrieve. 1000 is the default.

Return Parameter

Type	Description
CEvent	A CEvent object representing the Situation

↳ [getServices](#)

getServices()

Fetches a list of services from the database.

Request Argument

Name	Type	Description
limit	Integer	The maximum number of services to retrieve. 1000 is the default.

Return Parameter

Type	Description
CEvent	A CEvent object representing the Situation

↳ [getSituation](#)

getSituation()

Fetches a specified Situation from the database.

Request Argument

Name	Type	Description
situationId	Number	The Situation ID

Return Parameter

Type	Description
CEvent	A CEvent object representing the Situation

↳ [getSituationActions](#)

getSituationActions()

Returns activity for specified situations. Created by passing an object with information requested.

Request Arguments

Name	Type	Description
sitn_ids	JSON list	The Situation IDs
start	Number	Starting from which row should data be included in results
limit	Number	Limit for how many activities wanted in the output
actions	JSON list	List of action_codes of actions included in the return

Return Parameter

Type	Description
Native object	A Javascript object containing the activity for specified situations

Example

Input:

```
{
    "sitn_ids" : [1, 2, 3],
    "start" : 0,
    "limit" : 100,
    "actions" : [1, 14]
}
```

Return:

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

getSituationAlertIds

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]
    }
}
```

[getSituationIds](#)

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](#)

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.sourcefield 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"
    ]
}
```

[getSituationProcesses](#)

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](#)

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"
}
```

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
HTTP code	HTTP status or error code indicating request success or failure. For codes, see below .

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,
        ...
    ]
}
]

```

[getTeam](#)

getTeam

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure. For codes, see below .

Example 1 (team_id)

Curl Command:

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

```
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](#)

getTeamsForService

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure. For codes, see below .

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"
    ],
    "tags": [
    ]
}
]

```

[getTeamSituationIds](#)

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](#)

getThreadEntries()

Returns thread entries for the specified Situation. Threads are comments or 'story activity' on Situations (More information [here](#)).

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	Description
situationId	Number	The Situation ID
thread	String	The name of the thread
start	Number	The number of the first thread entry to return Optional
limit	Number	The maximum number of thread entries to return Optional

Return Parameter

Type	Description
Native object	A Javascript object containing details of the selected thread entries

Example

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
}
```

getUser

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

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](#)

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](#)

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](#)

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"  
}]
```

mergeSituations

mergeSituations()

Merges two or more Situations, superseding the originals if required, and returning the newly created Situation.

Request Arguments

Name	Type	Description
situationIds	Native array	A Javascript array containing the IDs of the Situations to merge
keepOriginals	Boolean	Determines what to do with the original Situations: true = keep the original Situations false = supersede the original Situations

Return Parameter

Type	Description
CEvent	A CEvent object containing the newly created Situation

moveSituationToCategory

moveSituationToCategory()

Move a Situation into a new category.

A category represents a type of Situation, indicating how it was created or its state. For more information, see [Alert and Situation Filters](#).

Request Arguments

Name	Type	Description
situationId	Number	The Situation ID
category	String	The name of the new category

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

moveSituationToQueue

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

↳ [removeAlertFromSituation](#)

removeAlertFromSituation()

Removes a specified Alert from a Situation.

Request Arguments

Name	Type	Description
alertId	Number	The Alert ID
situationId	Number	The Situation ID

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

↳ [removeSigCorrelationInfo](#)

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
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

↳ [resolveSituation](#)

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](#)

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](#)

setAlertCustomInfo()

Updates the custom information in the database for specified Alert.

This method can either be used with the alertInfoCEvent or with both the alertIDand customInfoMaparguments.

The mergeparameter 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	<p>The Alert ID.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Can be used alongside customInfoMapand mergebut not alertInfo. </div>
alertInfo	CEvent	<p>A CEvent containing alert_idand custo m_info attributes, the values of which will be used to replace the custom_info in the specified Alert.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Can be used alongside merge but not alertIdor customInf oMap. </div>
customInfoMap	Object	A map of name value pairs containing the new custom_infoinformation.
merge	Boolean	Determines what is done with the custom information: true = merge the existing data with the new

	<p style="margin: 0;">data</p> <p style="margin: 0;">false = replace the existing data with the new data.</p>
--	---

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail.

↳ [setAlertSeverity](#)

setAlertSeverity()

Sets the severity level for a specified Alert.

Request Arguments

Name	Type	Description												
alertId	Number	The Alert ID												
severity	Number	<p>The Alert's severity as an integer:</p> <table border="1" style="margin-left: 20px;"> <tr> <td>0</td> <td>Clear</td> </tr> <tr> <td>1</td> <td>Indeterminate</td> </tr> <tr> <td>2</td> <td>Warning</td> </tr> <tr> <td>3</td> <td>Minor</td> </tr> <tr> <td>4</td> <td>Major</td> </tr> <tr> <td>5</td> <td>Critical</td> </tr> </table>	0	Clear	1	Indeterminate	2	Warning	3	Minor	4	Major	5	Critical
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](#)

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

↳ [setSigCustomInfo](#)

setSigCustomInfo()

Updates the custom information in the database for specified Situation.

The Situation ID and new custom information are both contained in the `situationInfoCEvent`. The new custom information is contained in the `customInfoMap` object.

The `mergeparameter` determines whether to merge the new custom information data with existing data or replace it.

Request Arguments

Name	Type	Description
<code>situationId</code>	Number	The Situation ID
<code>customInfoMap</code>	Object	A map of name value pairs containing the new custom_infoinformation.
<code>merge</code>	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

↳ [setSituationProcesses](#)

setSituationProcesses()

Applies a list of `processes` (contained in the `processesJavascript` array) to a specified Situation.

Any other processes already associated with the Situation are removed.

Request Arguments

Name	Type	Description
<code>situationId</code>	Number	The Situation ID.
<code>processes</code>	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](#)

setSituationServices()

Applies a list of [external services](#) (contained in the services JavaScript array) to a specified Situation.

Any other services already associated with the Situation are removed.

Request Arguments

Name	Type	Description
situationId	Number	The Situation ID.
services	Native array	A JavaScript array containing the service names. If any services supplied do not exist in the database, they are created and assigned to the Situation.

Return Parameter

Type	Description
Boolean	Indicates if the operation was successful: true = success, false = fail

↳ [updateAlert](#)

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

↳ [updateSituation](#)

updateSituation()

Takes a Situation object and uses it to update the database and the MooMS 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

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 Cisco Crosswork Situation Manager 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 Cisco Crosswork Situation Manager as is
landing_page	String	Optional - The team default landing page. Leave empty to leave Cisco Crosswork Situation Manager as is
active	Boolean	Optional - False if the team is inactive, true if the team is active. Default to true. Leave empty to leave Cisco Crosswork Situation Manager as is
description	String	Optional - The team description. Leave empty to leave Cisco Crosswork Situation Manager as is
users	List of numbers or strings	Optional - The team users (either IDs or usernames). Leave empty to leave Cisco Crosswork Situation Manager 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](#)

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. 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
timezone	String	The user timezone
contact_num	String	The user phone number
session_expiry	Number	The number of minutes after which the user session will expire. Default to system default
competencies	JSON list	A list with the user competencies. Each competency should have have name or cid and ranking. That is, something like:

```
[{"name": "SunOS", "ranking": 40}, {"name": "SAP", "ranking": 50}, {"name": "EMC", "ranking": 60}]
```

teams	JSON list of numbers or strings	List of the user teams. The list should contains either the list of the teams ID or the teams name
-------	---------------------------------	--

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

Process

Description

The Process module allows you to run and control the execution of another process.

The Process module is available to load into any standard MooBot.

To use, at the top of a MooBot js file, define a new global object `proc` to load the Process module:

```
var proc = MooBot.loadModule('Process');
```

Create a new process with `create` and access methods to run the process with `arg`

Then run the process in one of two ways - either `run` in a separate child process of `moog_farmd`, or `runToExit` and only return when the process exits.

Stop processes running with `terminate`

These methods are detailed below.

Reference Guide

↳ [proc.create](#)

proc.create()

Defines a valid pathname to an executable file that you have permission to execute (or the user that started `moog_farmd` 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](#)

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](#)

proc.run()

Takes the object returned from createand runs the process in a separate child process of moog_farmd

Request Argument

Name	Type	Description
processObj	Object	The object returned from the createmethod

Return Parameter

Type	Description
Object	An object containing the process results

↳ [proc.runToExit](#)

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 createmethod

Return Parameter

Type	Description
Object	An object containing the process results

↳ [proc.terminate](#)

proc.terminate()

Stops the created processes running (causes the process object returned from createto be terminated)

Request Argument

Name	Type	Description
processObj	Object	The object returned from the createmethod

Return Parameter

Void - no value returned

Example

The following function runs an external tool toolNameusing 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();  
    }  
}
```

```

        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).

REST.V2

Description

REST (Representational State Transfer) and RESTful applications use HTTP requests to post data (create and/or update), read data (e.g. make queries), and delete data.

The REST.V2 MooBot

The REST.V2 MooBot module accesses an external RESTful API via HTTP or HTTPS, offering consistent usage between the available methods and customization of HTTP requests sent.

The REST.V2 MooBot module supports asynchronous operation (using Callback functions), to send a request without blocking the javascript code execution until the request is completed.

The REST.V2 MooBot module supports timeout (using the timeout property), to make the request fail after a specified time.

REST.V2 is available to load into any standard MooBot.

To use, at the top of a MooBot js file, define a new global object REST to load the REST.V2 module:

```
var REST = MooBot.loadModule('REST.V2');
```

Reference Guide

↳ [REST.sendGet](#)

REST.sendGet()

Sends a HTTP GET request to a third party (URL) with optional parameters:

Request Arguments

Name	Type	Description
url	String	The request URL. Mandatory

<parameters>	JSON Object	Optional parameters. See below
--------------	-------------	--------------------------------

Optional parameters

Name	Type	Description
params	String or Object	Either a String with the request encoded parameters or an Object with the parameters that will get encoded by the module
user	String	The user name for basic authentication
password	String	The password for basic authentication
encrypted_password	String	Encrypted version of password (encrypted using moog_encryptor)
disable_certificate_validation	Boolean	'true' to disable HTTPS server certificate validation by the MooBot
headers	Object	Any additional headers sent with the request
callback	Callback function	The request is sent asynchronously, returns null and the callbackfunction 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 successfunction is called only if the request was successful. See below
failure	Callback function	The request is sent asynchronously, returns null and the failurefunction 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

proxy	String or Object	<p>host, port, user, encrypted_password /password</p> <p>E.g. As an Object:</p> <pre> proxy: { host:"proxyhost", port:1223, user:"proxyuser", encrypted_passw ord:"2KctaEbJH /m8rz4WqgmXYZfd ripdIsku7fOFJWM 6YNA=" } //password: "unencrypted_pl </pre>
-------	------------------	---

```
ain_text_password"  
}
```

As an Object, you can either specify a Moog 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@proxyh  
ost: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 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.moogsoft.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.moogsoft.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"
        ]
    }
}
```

```

        ],
        "X-Powered-By": [ "PHP/5.3.10-
          1ubuntu3.10"
      ]
    }
  }
}

```

[REST.sendPost](#)

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 _typeis “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 nulland the callbackfunction is called regardless of the success or failure of the request. See below
success	Callback function	The request is sent asynchronously, returns nulland the successfunction is called only the request was successful. See below
failure	Callback function	The request is sent asynchronously, returns nulland the failurefunction 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
proxy	string or object	<p>host, port, user, encrypted_password /password</p> <p>E.g. As an Object:</p> <pre>proxy:{ host:" proxyhost", port:1223, user:" proxyuser", encrypted_passw ord:"2KctaEbJH /m8rz4WqgmXYZfd ripdIsku7fOFJWM 6YNA=" //password: "unencrypted_pl ain_text_password" }</pre> <p>As an Object, you can either specify a Moog encrypted password or a plain text password, specifying both will favour the encrypted_password value.</p> <p>Or, as a String, where format is <user>:<password>@<host>:<port></p> <pre>proxy: "proxyuser: passw0rd@proxyh ost:1223"</pre> <p>Only plain text passwords are supported in the String format</p>

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 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 accesses DuckDuckGo and searches for 'Moogsoft'.

Synchronous request:

```
var rc = REST.sendPost('https://api.duckduckgo.com/',  
{q:'Moogsoft', format:'json', pretty:1});
```

Asynchronous request:

```
REST.sendPost({url: 'https://api.duckduckgo.com/',  
              body: {q:'Moogsoft', 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\" : \"\",  
               \"ImageWidth\" : 0, ... : \"\" }",  
  "status_msg": "OK",  
  "headers": {  
    "Content-Type": "application/json",  
    "Content-Length": "110",  
    "Date": "Mon, 12 Jun 2017 11:11:11 GMT",  
    "Server": "nginx/1.11.6 (Ubuntu)"  
  }  
}
```

```

    "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](#)

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 _typeis "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 nulland the callbackfunction is called regardless of the success or failure of the request. See below
success	Callback function	The request is sent asynchronously, returns nulland the successfunction is called only the request was successful. See below
failure	Callback function	The request is sent asynchronously, returns nulland the failurefunction 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, successand 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.

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, if we wanted to send the entire alert custom_info object as part of a url-encoded body, we would do the following:

```
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](#)

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 callbackfunction 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 successfunction is called only if the request was successful. See below
failure	Callback function	The request is sent asynchronously, returns null and the failurefunction 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, successand 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.

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": "{\"remoteId\": 33, \"weight\": 0.8240487528964877,  
    \"location\": {\"latitude\": 147.3387699946761, \"longitude\":  
    -7.957067163661122},  
    \"status_msg\": \"OK\",  
    \"headers\": {  
        \"Transfer-Encoding\": [  
            \"chunked\"\br/>        ],  
        \"Connection\": [ \"keep-  
            alive\"\br/>        ],  
        \"Date\": [  
    }
```

```
        ],
        "Content-Type": [
            "application/json"
        ]
    }
}
```

REST.send

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 Moog server:

Request

```
var rc = REST.send({method: 'HEAD', url: 'http://www.moogsoft.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"
        ]
    }
}
```

```

        "text/html"
    ],
    "X-Powered-By": [ "PHP/5.3.10-
        1ubuntu3.10"
    }
}

```

Proxy Example

Proxy Example

As an example, you could modify the SituationMgr.js in order to send an updateSituation message via REST.V2 through a proxy server:

```

function updateSituation(situation)
{
    var sig_id = situation.value("sig_id"); logger.warning("Update Situation Processed:
        " + sig_id);
    doPOST(sig_id);
}

```

insert the proxy block into the REST.sendPost({...}) function as below:

```

function doPOST(sig_id)
{
    var request = REST.sendPost({ url:"http://surveilanceserver_84:9090/reportAntiSoc",
        params: {
            crime: "Graffiti"
        },
        proxy: {
            host: "proxyserver", port :
            3128,
            user : "username",
            encrypted_password:"zm0lxjTGiAhp6LrpM49+kr4SDtHj
/fq16+i+hD1MG4c="
        },
        callback: function(response, request)
        {
            if (response.success) {
                logger.warning("4764 CALLBACK SUCCESS
                ("+sig_id+) RESPONSE - ("+response.status_code +" - "+response. response+) REQUEST - "+
                JSON.stringify(request));
            } else {
                logger.warning("4764 CALLBACK FAILURE
                ("+sig_id+) RESPONSE - ("+response.status_code +" - "+response. response+" -
                "+response.status_msg+) REQUEST - " +request.status_code
                + " " + request.response + " " + request.status_msg);
            }
        }
    });
}

```

```
});  
logger.warning("4764 POST REQUEST SENT FOR "+sig_id+" ...");
```

```
}
```

Some alternative methods of using the proxy functionality are below:

```
1.) proxy: "proxyuser:passw0rd@proxyhost:1223"  
2.) proxy: "proxyhost:1223"  
3.) proxy: {  
    host: "proxyhost", port:  
    1223  
}
```

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](#)

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 "< and >"</tag>
```

↳ [utilities.unescapeXML](#)

utilities.unescapeXML()

Unescapes an XML string.

Name	Type	Description
value	String	The string to unescape.

Example

```
var escapedXML = '<tag>my content requires "< and >"</tag>';  
var unescapedXML = utilities.unescapeXML(escapedXML);
```

The variable unescapedXML now contains:

```
<tag>my content requires "< and >"</tag>
```

↳ [utilities.xmlToJson](#)

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>' +

```

```

var alerts = utilities.xmlToJson(xmlExample);

```

The variable `alertsnow` 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](#)

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>

```

Moolets

Introduction

A Moolet is an intelligence module that is used to perform specific services in Cisco Crosswork Situation Manager. Some Moolets have an accompanying MooBot, a Javascript file that controls or customizes the behavior of a Moolet.

There are four ways in which a Moolet can be triggered by events within Moogfarmd:

Trigger	Description
process_output_of	the Moolet is told to listen for events from another named Moolet (used to chain Moolets together to form an automated workflow pipeline)
mooms_event_handler	the Moolet can now listen for events on the message bus, e.g. actions triggered by a user or within another farmd
standalone_moolet	the Moolet can listen for events generated by other Moolets within the same moogfarmd instance without being part of a process_output_of chain
scheduler	a unique Moolet type that allows time based task execution

Click the links below for more information about the available Moolets:

Alert Builder Moolet

The Alert Builder Moolet assembles from Events, sent by the LAMs across the MooMS bus, the Alerts that are visible through the Alert View in the User Interface (UI). The Alert Builder Moolet is also responsible for:

- Updating all the necessary data structures. For example, when a duplicate Alert arrives in the system with the same signature
- Ensuring copies of the old Alert state are stored in the snapshot table in moogdb, relevant events are created and the old Alert record is updated to reflect the new events arriving into the system

Alert Builder Configuration Walk-through

The behaviour of the Alert Builder is defined in the moog_farmd configuration file in a section titled AlertBuilder.

```
{
    name:"AlertBuilder",
    classname:"CAlertBuilder",
    run_on_startup:true,
    moobot:"AlertBuilder.js",
    event_streams:[
        "AppA"
    ],
    threads:4, metric_path_moolet:true,
    events_analyser_config:"events_analyser.conf",
    priming_stream_name:null, priming_stream_from_topic:false
}
```

AlertBuilder only contains a few parameters: name, classname, and run_on_startup are shared with other Moolets. See the table below for more information:

Parameter	Description
name	name is hardcoded and should never be changed from AlertBuilder
classname	the classname, CAlertBuilder, is hardcoded and should never be changed
	by default, run_on_startup is set to true, so that when moog_farmd starts, it automatically creates an instance of the Alert

run_on_startup	Builder. In this case you can stop it using farmd_ctrl
events_analyser_config	allows configurations for tokeniser rules to be specified on a moolet-by-moolet basis. If no config is specified, the system default is used
priming_stream_name	the stream name under which the events_analyser was run in order to calculate alert entropies. If set to null all alerts are factored into the entropy calculation
priming_stream_from_topic	if set to true the priming_stream_name is extracted from the event's stream. If set to false the stream to use is the value configured in priming_stream_name
moobot	moobot specifies a JavaScript file found in \$MOOGSOFT_HOME /moobots , which defines the AlertBuilder Moobot, which creates Alerts
event_streams	<p>a list of sub event streams, which the Moolet in this instance of farmd will process. The LAMs can be configured to send events on different sub streams. farmd, as specified in the Alert Builder configuration, then decides whether or not to process them. If MOOG runs multiple farmd's, you can have different event sub streams being processed by different Alert Builder Moolets</p> <p>by default, you can comment out event_streams, or provide an empty list; subsequently, the Alert Builder will process every event that is published on the default /Events topic on the Mooms bus</p> <p>you configure the Alert Builder Moolet by giving it a list of strings, for example, <code>["App A", "App B"]</code>. The result is that the Alert Builder will listen for events published on /Events/AppA, as well as, /Events/AppB and process that data. Importantly, in this example, events published to /Events or any other substream are ignored. You can have farmd's that are processing completely separate event streams, or, multiple farmd's that process some different event streams and some common event streams. You would do this when some of the Alerts are common to all of the applications that are being processed, but some are specific only to a given application. In this way, you would cluster Alerts separately for each application, as the Sigaliser only processes alerts from its upstream Alert Builder Moolet</p> <p>for example, if you have two separate applications that share the same network infrastructure: in farmd 1, you can have as the event streams, application A and networks, and, in farmd 2, you can have application B and networks. So you can detect Alerts and then create Situations that are relevant for just application A; however, if there is common networking infrastructure and problems occur with network failures, you will get those clustered into Situations, and similarly for application B</p>
threads	the number of threads in the Alert Builder is chosen to match the event rate experienced by the system and allows time Alert creation. By default, the Alert Builder is only run in single threaded mode

AlertBuilder.js

Most of the activity of the Alert Builder is undertaken in the Moobot, **AlertBuilder.js**, associated with the Alert Builder Moolet. The JavaScript function, `newEvent`, is called when the Alert Builder Moolet processes an event:

```
events.onEvent ("newEvent", constants.eventType("Event")).listen();
```

- `newEvent` contains a call to create an Alert
- The newly created Alert is broadcast on the Mooms bus

The Alert Builder Moobot is explained in full in the Moobot and Moobot Language documentation.

Alert Rules Engine

The Alert Rules Engine (ARE) applies business logic to event processing in Cisco Crosswork Situation Manager. You can define rules in ARE to hold alerts for a period of time, identify missing events or change the state of events.

For example, common uses of ARE include:

- **Link Up-Link Down** - delays an alert to see if a link recovers.
- **Heartbeat Monitor** - detects any missing network health signals.
- **Closing Events** - close events of a particular type or severity.

The ARE controls event processing by placing alerts in different virtual buckets called [Action States](#). These determine the period of time an alert is held for or if it should be passed to the next Moolet or Sigaliser algorithm in the chain. You define when an alert moves from one Action State to another using [transitions](#), a configurable set of conditions and filters. Transitions moving alerts from one state to another results in the following:

- Alerts moving from one Moolet to the next moolet in the chain
- Alerts being passed to a Sigaliser and clustered into Situations.

Configure the Alert Rules Engine

You can configure Alert Rules Engine in \$MOOGSOFT_HOME/config/moog_farmd.conf using the following parameters:

run_on_startup

Determines whether Alert Rules Engine runs when Cisco Crosswork Situation Manager starts up.

Type: Boolean

Default: false

persist_state

Enables Alert Rules Engine to save its state for [High Availability](#) systems. When a failover occurs, the standby moogfarmd continues processing events from where the primary stopped.

Type: Boolean

Default: false

metric_path_moolet

Determines whether or not Cisco Crosswork Situation Manager factors ARE into the [Event Processing](#) metric for [Self Monitoring](#).

Type: Boolean

Default: false

moobot

The name of the Alert Rules Engine JavaScript source. The file must reside at \$MOOGSOFT_HOME/bots/moobots. The default, AlertRulesEngine.js, provides the standard modules. You can customize it to meet your needs.

Type: String

Default: "AlertRulesEngine.js"

process_output_of

Defines the input source for the Alert Rules Engine. This determines the Alert Rules Engine's place in the event processing workflow.

Type: List

Default: "MaintenanceWindowManager"

The default Alert Rules Engine parameter values are as follows:

```
{  
    name : "AlertRulesEngine", classname  
          : "CAlertRulesEngine",  
    run_on_startup : false,  
    persist_state : false,  
    metric_path_moolet : true,
```

```
moobot : "AlertRulesEngine.js",
# Configuration for Netcool LAM
# moobot : "AlertRulesEngineNetcool.js",
process_output_of : "MaintenanceWindowManager"
'
```

The classname is hardcoded and should not be changed.

After you have configured ARE to meet your needs, save the changes and restart the `moog_farmd` service:

```
service moogfarmd restart
```

Define Action States and Transitions

After you have configured the Alert Rules Engine, set up Action States and transitions in the Cisco Crosswork Situation Manager UI under **Settings > Automation**:

- **Action States** - determine the length of time Cisco Crosswork Situation Manager retains alerts before forwarding them to a Sigaliser or closing them.
- **Transitions** - defines the set of conditions an alert must meet before it moves from one state to another in the Alert Rules Engine. Higher priority transitions take precedence over those with lower priorities.

For more information see [Action States and Transitions](#).

The initial state for all alerts is the 'Ground' state. After an alert enters 'Ground' state, Cisco Crosswork Situation Manager transitions it to another state or forwards it to a Sigaliser. If the Action State has a 'Remember Alerts For' set to a positive number then Cisco Crosswork Situation Manager retains an alert in that state for this period of time.

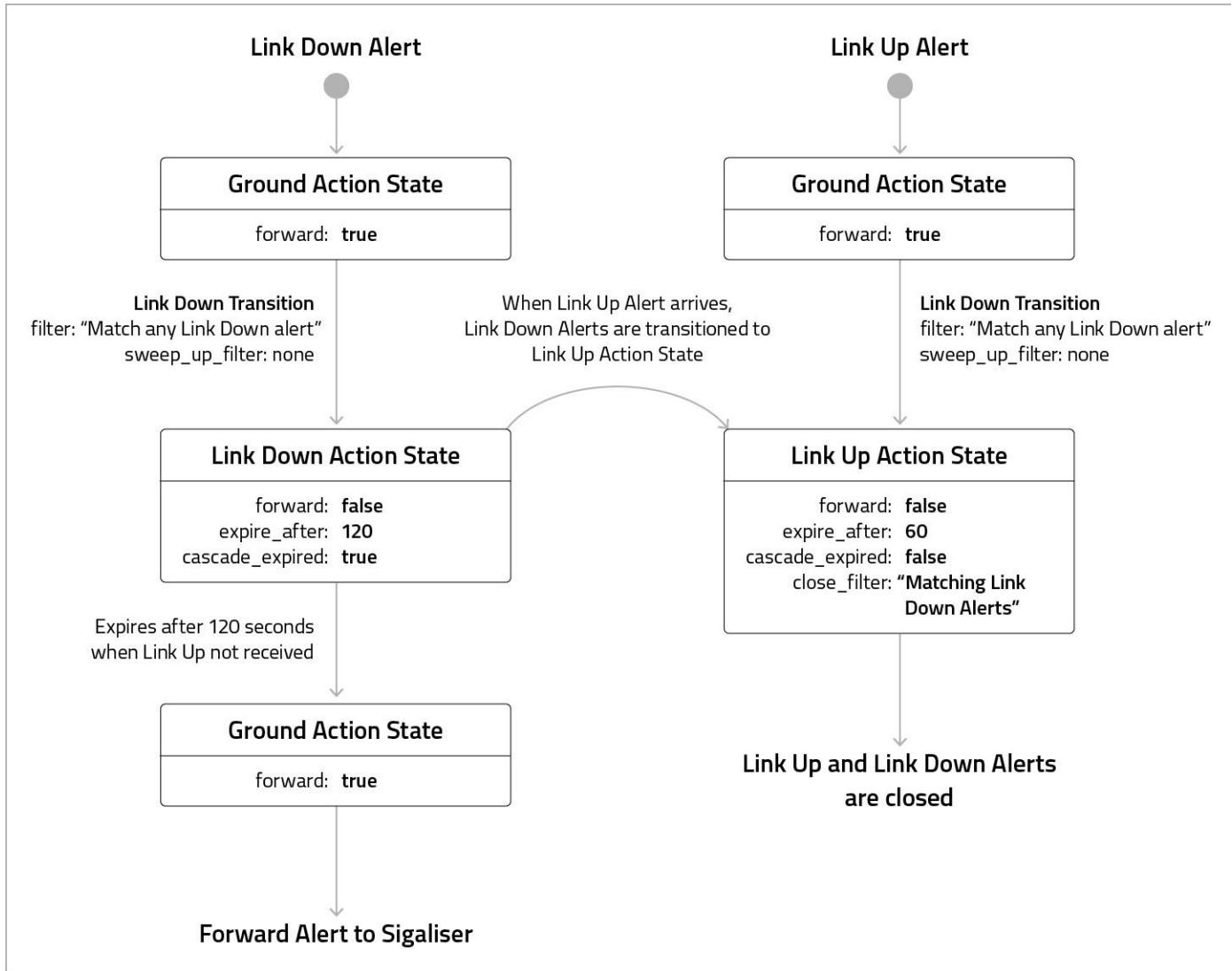
If you enable 'Cascade on Expiry' and nothing happens to an alert within that period, Cisco Crosswork Situation Manager returns it to 'Ground' state before forwarding it to a Sigaliser. This is because the 'Ground' state has "Forward Alerts" enabled. If an alert does not match any transitions, Cisco Crosswork Situation Manager does not return it to 'Ground' state and it is closed.

Action States are not enabled until you have defined a transition.

Link up/Link Down Example

This example demonstrates how to configure Alert Rules Engine so that when a link-down event arrives at `moog_farmd`, Cisco Crosswork Situation Manager holds it for a period of time to provide an opportunity for the link-up to arrive. If nothing arrives, Cisco Crosswork Situation Manager forwards it to a Sigaliser.

If the link-up arrives, the system closes and discards both alerts without sending anything to the Sigaliser. This ensures neither the link-down or link-up events appear in Situations.



To try out this example, set up the following:

1. Create three Action States: 'Ground' (default), 'Link Up' and 'Link Down'.
2. Create two transitions: 'LinkDown Transition' and 'LinkUp Transition'.

In this scenario, if a 'Link Down' alert arrives at the ARE and no 'Link Up' alert arrives within 120 seconds, then the 'Link Down' alert returns 'Ground State' and ARE passes it to a Sigaliser.

Heartbeat Monitor Example

For instructions on how to set this up see [Heartbeat Monitor](#).

Heartbeat Monitor

You can configure the [Alert Rules Engine](#) in Cisco Crosswork Situation Manager to detect missing heartbeat events from monitoring tools such as CA Spectrum and Microsoft SCOM. Both of these tools send regular heartbeats to indicate normal operation.

After you configure the Alert Rules Engine (ARE), Cisco Crosswork Situation Manager creates a Situation when an event source does not send a heartbeat after a given time period. The ARE holds each heartbeat alert for a period of time, subsequent alerts from the same heartbeat source reset the timer. If the timer expires, a heartbeat has been missed and the alert is forwarded to a Sigaliser.

Before you Begin

Before you set up the heartbeat monitor in Alert Rules Engine, ensure you have met the following requirements:

- You have an understanding of Alert Rules Engine, Action States and transitions. See [Alert Rules Engine](#).
- You can identify heartbeat alerts in the integration by description, class or another configurable field. These must be specific, regular events that arrive consistent intervals to indicate normal operation. If these are not available the Heartbeat Monitor will not work.

- You have edited the alerts so they contain the same attribute (via the integration source or through enrichment). In the example below, 'class' is set to 'heartbeat'.

Create a Heartbeat Monitor

To create a heartbeat monitor in Alert Rules Engine, follow these steps:

1. Edit \$MOOGSOST_HOME/bots/mobots/AlertRulesEngine.js.
2. Add the heartBeatSeverity exit action to the AlertRulesEngine.js. This function changes the alert severity to critical and ensures alerts that are closed (see [Status ID Reference](#)) are not forwarded to the Cookbook:

```
// Checks the state of the alert. If the alert's state is 9, the status ID for 'closed', it is not
forwarded.

function heartBeatSeverity(alert,associated) {
    var currentAlert = moogdb.getAlert(alert.value("alert_id")); if ( currentAlert &&
    currentAlert.value("state") !== 9 ) {

// Adds customInfo list displaying times of all missed heartbeats. var
    customInfo=currentAlert.getCustomInfo();
    if ( !customInfo ) { customInfo={};
        customInfo.missedHeartbeats=[];
    }
    if ( !customInfo.missedHeartbeats ) {
        customInfo.missedHeartbeats=[];
    }

// Changes severity of heartbeat alerts to 5, the severity level for 'critical', adds the description
'MISSED' and determines where the alerts are sent.

    var now = new Date(); customInfo.missedHeartbeats.push(now.toString());
    moogdb.setAlertCustomInfo(currentAlert.value("alert_id"),
    customInfo,false); moogdb.setAlertSeverity(currentAlert.value("alert_id"),5); var alertDescr
    = currentAlert.value("description"); alertDescr =
    alertDescr.replace(/(OK|SLOW)/,"MISSED");
    currentAlert.set("description",alertDescr); moogdb.updateAlert(currentAlert);
    currentAlert.forward("HeartbeatCookBook");
}
}
```

3. Navigate to [Settings > Action States](#) in the Cisco Crosswork Situation Manager UI.
4. Create a new Action State called "Heartbeat" as follows:

Setting Name	Input	Value
Name	String	Heartbeat
Remember alerts for	Integer (seconds)	30*
Cascade on expiry	Boolean	True
Exit Action	String	heartBeatSeverity

The 'Remember alerts for' setting is the timer. Set this to two or three times your heartbeat interval time.

5. Go to Transitions and set up a transition to move your heartbeat alerts to the 'Heartbeat' State. Configure the settings as follows:

Setting Name	Value
Name	Heartbeat
Priority	10
Active	True
Trigger Filter	(type = "heartbeat") AND (((agent = "SPECTRUM") OR (agent = "SCOM")) OR (agent = "MONITOR1")) OR (agent = "MONITOR2"))
Start State	Ground
End State	Heartbeat

Edit the 'Trigger Filter' to meet your requirements. In this example, the transition is triggered by alerts with the type of 'heartbeat' and that come from either 'SPECTRUM' or 'SCOM' or 'MONITOR1' or 'MONITOR2':

Trigger Filter

X
No clause selected

```

(type = "heartbeat") AND (((agent = "SPECTRUM") OR (manager = "SCOM")) OR (agent = "MONITOR1")) OR (agent = "MONITOR2"))

```

AND
OR
NOT
DELETE
EMPTY FILTER

CANCEL
DONE

6. Run this command to open moog_farmd.conf in vi:

```
vi /usr/share/moogsoft/config/moog_farmd.conf
```

7. Ensure Alert Rules Engine is enabled. To do this set run_on_startup to true.
 8. Add heartbeat Cookbook for heartbeat alerts. This only work with these alerts:

```
{
  name : "HeartbeatCookBook",
  classname : "CCookbook",
  run_on_startup moobot : true,
  process_output_of : "Cookbook.js",
  # Algorithm
}
```

```

membership_limit      : 5,
scale_by_severity : false, entropy_threshold :
0.0, single_recipe_matching : false, recipes :[
# Any heartbeat class for the same agent.
{
  chef           : "CValueRecipe",
  name          : "ScomHeartbeatErrors",
  description   : "SCOM Heartbeat: Missing heartbeat",
  recipe_alert_threshold : 0,
  exclusion     : "state = 9",
  trigger_rate  : "class = 'heartbeat' AND agent = 'SCOM'".
  min_sample_size : 0,      # Given in events per minutee
  max_sample_size : 5,
  matcher        : 10
    : { components: [ { name: "agent", similarity:
  1.0 } ] }
}
,
{
  chef           : "CValueRecipe",
  name          : "ScomHeartbeatChange",
  . . .
  . . .
  recipe_alert_threshold : 0,
  exclusion     : "state = 9",
  trigger       : "class = 'heartbeatRoleChange' AND agent =
  'SCOM'",
  rate min_sample_size : 0,      # Given in events per minutee
  max_sample_size : 5,
  matcher        : 10
    : { components: [ { name: "agent", similarity:
  1.0 } ] }
}
],
  cook_for      : 20000
}

```

9. Save the changes and restart moogfarmd:

```
service moogfarmd restart
```

After the heartbeat monitor configuration is complete, heartbeat alerts should start to arrive in Cisco Crosswork Situation Manager.

Heartbeat Monitor Process

The process flow for a heartbeat alert is as follows:

- Heartbeat alert arrives at the Alert Rules Engine.
- The alert is transitioned from 'Ground' to 'Heartbeat' action state and starts the timer.
- The alert sits in the 'Heartbeat' state waiting for the timer to expire.
- Any subsequent heartbeat alert resets the timer.

- If the timer expires, the exit action changes the alert severity to '5' (critical) and cascades it to 'Ground' state.
- Any subsequent heartbeat updates the severity to '0' (clear) and restarts the timer.
- You could also add an entry action to close any missed heartbeat situations the event is part of.

This example also updates the alerts with the times of the missing heartbeats for an easy audit trail.

Status ID Reference

The states of alerts and Situations is determined by their status ID.

The different status_id values are as follows:

```
mysql> select * from status;
+-----+-----+
| status_id | name
+-----+-----+
|        4 | Acknowledged
|        6 | Active
|        3 | Assigned
|        9 | Closed
|        7 | Dormant
|        1 | Opened
|        8 | Resolved
|       10 | SLA Exceeded
|        5 | Unacknowledged
|        2 | Unassigned
+-----+-----+
10 rows in set (0.00 sec)
```

Classic Sigaliser

- Introduction
- Basic concepts
- Sigaliser configuration walk-through

Introduction

The Sigaliser Moollet, also known as Sigaliser Classic, is where in [Event processing](#), Alert streams from the Alert Builder or the Alert Rules Engine are converted into Situations.

The Sigaliser is self-contained and has no MooBot. It takes every occurrence of an Event in an Alert stream and uses matrix factorisation algorithms to identify clusters of Alerts that are temporally correlated identifying underlying service outages or Situations. The Sigaliser then updates its own internal knowledge of the stores of the Situations and the Cisco Crosswork Situation Manager database before putting updates out on the Mooms bus.

Basic concepts

There are a number of parameters in **moog_farmd.conf** which allow you to tune the type of Situations created by the Sigaliser. Generally, the types of Situations created for a given set of alerts are dependent on the rate of occurrence of alerts. You correspond by adjusting the resolution of the window of the Sigaliser parameters to try and match the activity.

The algorithms work by spotting signature scatter pattern of Alerts with in a time period. Firstly, how many optimal clusters there are, which should correspond to the number of current, active, service threatening outages in the given window that the Sigaliser operates on. Secondly, it then optimally factorises it down into individual groups, which Cisco Crosswork Situation Manager calls Situations. Once you have a Situation, a Situation Room is created in the MOOG database, and you are notified through the Situation View in the User Interface.

The algorithm is run in semi real-time and is triggered by either:

- A fixed polled time period
- A single time slice being filled up, the width of which is set by the resolution parameter in the configuration. For example, the first alert that arrives after the current slice has been filled will trigger the Sigaliser to run its algorithms

Sigaliser configuration walk-through

The behaviour of the Sigaliser is defined in the moog_farmd configuration file in a section titled Sigaliser. In general, the following parameters can be configured to either: produce more Situations with fewer alerts, or, fewer Situations with more alerts. The consequence of having more Situations with fewer alerts is that the same underlying outage could be split across multiple Situations. Fewer Situations with more alerts results in the same Situation containing alerts from multiple service outages. The process of tuning the Sigaliser parameters leads to an optimal configuration, where, Situations sharply reflect the state of the managed systems. Cisco Crosswork Situation Manager refers to Situations being "sharp" and well "resolved" when the parameters give you the best fit of Situations to service outages.

Sigaliser contains a number of parameters. name, classname, and run_on_startup are shared with other Moolets.

```
{  
    name : "Sigaliser", classname :  
        "CSigaliser", run_on_startup : false,  
        #process_output_of : "AlertRulesEngine",  
        process_output_of : "AlertBuilder",
```

name

name is hardcoded and should never be changed from Sigaliser.

classname

The classname, CSigaliser, is hardcoded and should never be changed.

run_on_startup

By default, run_on_startup is set to false, so that when moog_farmd starts, it does not automatically create an instance of the Sigaliser. In this case you can start it using farmd_ctrl.

Undertaking the sigalising

The next two parameters in the Moolet direct which output should be processed:

process_output_of

process_output_of informs the Moolet to process the output of the Alert Builder or Alert Rules Engine. Usually the Sigaliser connects directly to the AlertBuilder, the AlertRulesEngine only being used if automations are desired prior to Situation resolution. The Sigaliser can have only one input.

Algorithmics

The Sigaliser runs the matrix factorisation algorithms, the parameters for which are identified in the configuration below:

```
# Algorithm time_compression :  
true, alert_threshold : 2,  
membership_limit : 3,  
sig_similarity_limit : 0.7,  
sig_alert_horizon : 0.5,  
scale_by_severity : false,  
entropy_threshold : 0.0,
```

time_compression

If set to true, the algorithm will ignore any empty time buckets in the Sigaliser calculation. If set to false, it will include the empty time buckets. Cisco recommends for low data-rates you should set time_compression to true, and for normal data-rates, time_compression should be set to false.

You only require time_compression in scenarios where the data rate is very low when compared to the values of window and resolution. In certain low data-rate scenarios it is possible for a window or resolution to contain no alerts. For example if the data rate is two alerts per

hour and the window is 15 minutes, on average, some of the time buckets in any Situation calculation will be empty. When time_compression is true empty time-buckets are removed from the calculation, but the total number of buckets used in the calculation remains the same.

alert_threshold

Defines the minimum number of alerts that a Situation can contain. So, increasing the alert_threshold will reduce the total number of Situations. Cisco recommends an alert_threshold of 2.

alert_threshold can be used in conjunction with small values of membership_limit to produce a smaller number of Situations, each of which has more alerts.

membership_limit

The Situation creation process contains multiple steps, including a resolution and merging step. During the merging phase, the raw Situations from the factorization calculation are compared and merged with the currently active Situations. This detects when a detected Situation is either novel or an evolution in time of an existing Situation.

membership_limit restricts the number of Situations that an alert can appear in. As Situations get merged with each other over time, it is possible for an alert to appear in more Situations than are defined by membership_limit. Changing the value of membership_limit does not have a large impact on the total number of Situations but does change the distribution of the number of alerts in each Situation.

Decreasing the membership_limit results in fewer Situations with more alerts and more Situations containing a small numbers of alerts. Whereas, increasing membership_limit results in, more Situations with a greater number of alerts and fewer Situations containing a small numbers of alerts. Therefore, the optimal value seems to be between one and five, with a recommended membership_limit of three.

sig_similarity_limit (Jaccard Similarity Coefficient)

A measure of the similarity between two Situations before they are merged together. The value is the [Jaccard Similarity Coefficient \(JSC\)](#) defined as the ratio of shared Alerts between two Situations to total unique Alerts in both Situations.

For example, if Situation1 & Situation 2 share two common Alerts, each Situation has one unique Alert:

$$\text{JSC} = 2 \text{ (common Alerts)} / [1 \text{ (unique to Situation 1)} + 2 \text{ (common to both)} + 1 \text{ (unique to Situation 2)}] = 2/(1+2+1) = 2/4 = 0.5.$$

Reducing the similarity index will reduce the total number of Situations. Smaller values increase the likelihood of Situations being merged together, as they have to share fewer Alerts in common to be viewed as the same Situation. Conceptually, JSC values less than 0.5 are hard to justify as grounds for merging, so should be used with care. Cisco recommends a sig_similarity_limit of 0.7.

sig_alert_horizon

When the Sigaliser algorithm initially identifies a Situation, it will contain alerts that are more representative of the Situation than others. This parameter, which takes the value between 0.0 and 1.0, allows you to provide a cut off for membership based upon the highest significant alert in the cluster. If you set this value to be 0.5, for example, only alerts that have a "significance" for the Situation that is more than half of the most significant alert in the Situation will be included. 0.5 is the default value.

entropy_threshold

The value of this parameter is the minimum entropy that an alert must possess to be included in the Sigaliser calculation. Any alert that arrives at the Sigaliser with entropy below this value will never be included in a Situation. It has a value between 0.0 and 1.0 and has a default of 0.0 which means every alert will be processed.

scale_by_severity

scaleBySev allows you to bias MOOG so that high severity alerts are treated as having higher entropy. If you had the same alert arrive with a critical severity, versus a minor severity, you would give the critical severity the higher entropy than the minor severity. This scaling is done as the severity constant number divided by the maximum severity (5). So in the case of critical, you get all of the entropy and in the case of minor, you get three fifths of the entropy. In the case of clear you would get an entropy value of 0.0.

Triggers and Time Buckets

The algorithm is run incrementally as Events are ingested, as such Situations are produced and updated in real-time. There are two ways to trigger the algorithm: using a time interval or using the rate of the Event stream.

```
# Triggers sig_on_bucket :  
true, sig_interval : 100,  
max_backlog : 1000000, # Time  
Buckets resolution : 120,
```

```
        window : 90  
    }
```

The optimal trigger for production should be `sig_on_bucket=true`, provided this ensures satisfactory Situation accuracy and that Situations are being regularly updated. `sig_on_bucket` can also simulate real-time behavior using historical data.

When Situations are not being updated regularly enough, configure `sig_on_bucket=false` and set `sig_interval` to a value no more than half of the real-time size of the window.

In a production environment, set `max_backlog` to a high value to avoid triggering the Sigaliser between timed executions. This parameter will cause the algorithms to run if the number of Events that arrive before either a scheduled execution, or a bucket being filled is above this value. It should be used with care and only when you have an environment where the event rate is highly variable.

sig_on_bucket

If set to true, the Sigaliser will run whenever a new time bucket occurs. Depending upon the data rate, this has the effect of executing the Sigaliser after every defined number of "resolution" seconds.

`sig_on_bucket = true` deactivates both the `sig_interval` and `max_backlog` triggers.

sig_interval

Executes the Sigaliser algorithm every defined number of seconds, in the example above, every 100 seconds.

`sig_interval` and `max_backlog` do not override each other; consequently, it is possible for the Sigaliser to be executed more frequently through the `sig_interval` value.

max_backlog

Executes the Sigaliser if the number of defined Alerts are received since last execution, in the example above, the Sigaliser is executed after 1,000,000 Alerts are received.

resolution

The duration, in seconds, for each bucket of time that the event stream is divided into. A high value for the resolution will result in Situations that are less "sharp" in time, as the wider the bucket the more likely that alerts from disconnected outages will occur in the same bucket, and potentially in the same Situation.

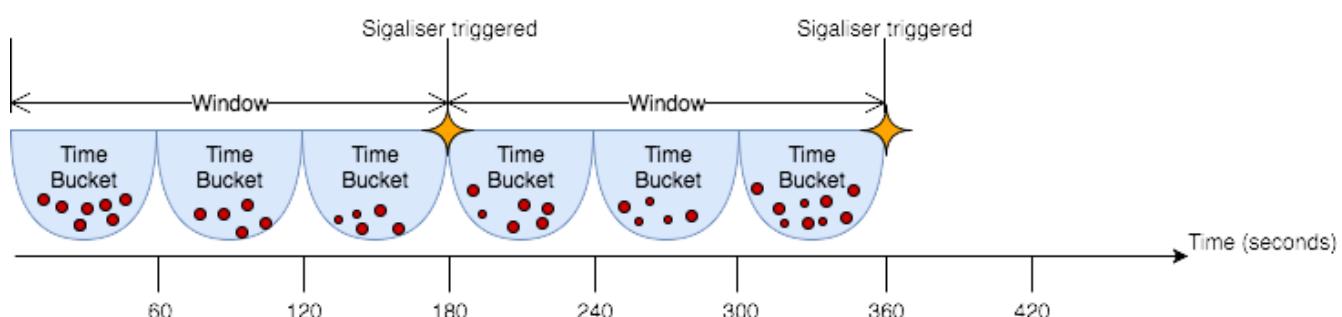
window

The number of time-buckets to include in the calculation. The width of the window should be chosen to match the average time period over which outages typically evolve. The total amount of time considered in any Sigaliser calculation is `window` multiplied by the `resolution`.

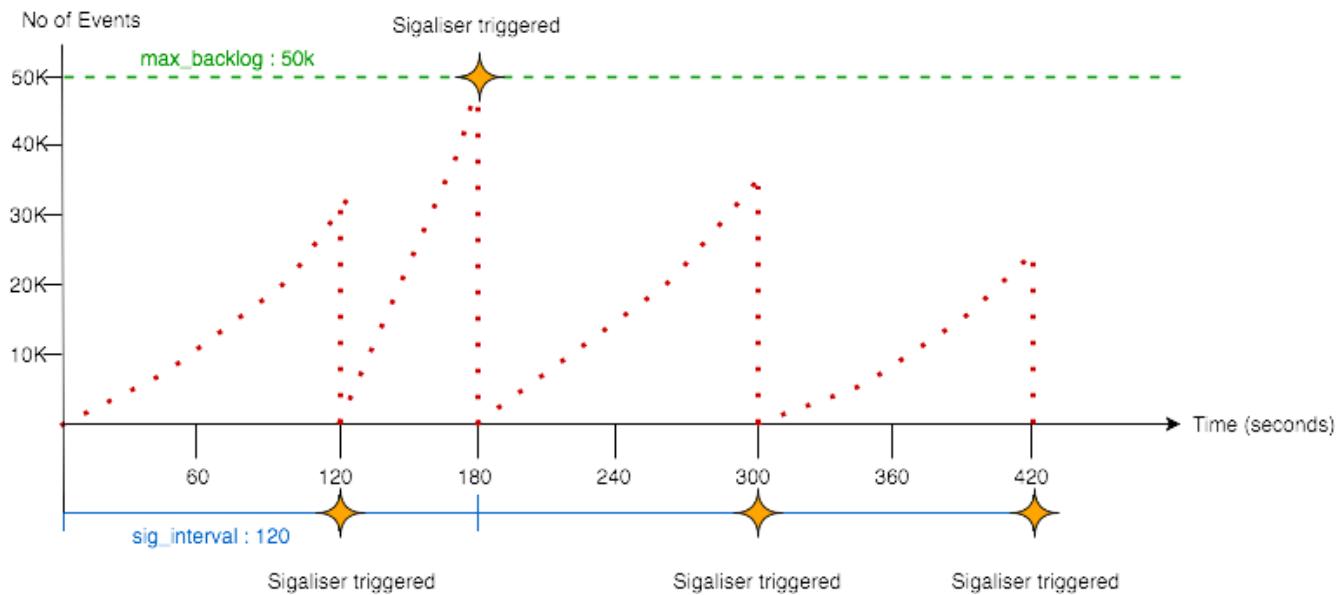
In general, for a high data rate you would use a smaller `resolution` and `window` than for a low data rate. For a fixed data rate, a smaller `resolution` will generally result in more Situations.

Diagrams

The diagram below illustrates how a sigaliser can be triggered every 180 seconds if 'sig_on_bucket' is set to 'true', the time bucket resolution is set to '60' and the window is '3':



The diagram below illustrates how a sigaliser can be triggered if 'sig_interval' is set to 120 seconds and if 'max_backlog' is set to 50,000 Events:



Empty Moolet

- Introduction
- Configuration
- AlertMgr

Introduction

The Empty moolet enables Cisco Crosswork Situation Manager integrators to intercept and handle MooMS events without impacting upon the existing Cisco Crosswork Situation Manager Event flow logic and processing. This makes available a canvas upon which individual pieces of integration logic can be superimposed.

For example, to integrate with an incident management system, such as ACMEIncidentManager, the Empty Moolet must:

- Listen to NewThreadEntry Events (topic on MooMS is /sig/thread/entry/new) and SigStatusEvents (topic on MooMS is /sig s/status topic). Other handlers will follow in due course
- Interrogate the Events to select only those in which the incident management system has registered an interest via the Graze API addSigCorrelationInfo request
- Filter out those Events which were originated by the incident management system via the Graze API to avoid sending duplicate information
- Extract relevant information from the Event including the incident management system entity reference
- Send the information to the incident management system via the REST.V2 MooBot module which supports the sending of simple RESTful POST requests using basic HTTP authentication

The above example is only one way of integrating Cisco Crosswork Situation Manager with other systems. Each integration will be dependent upon the individual use cases and systems being integrated.

The Empty Moolet can also be used in general augmentation of Alert and Situation details.

Configuration

The Empty Moolet is simply designed to take messages off the MooMS bus according to message type and pass them to a Moolet. The configuration is correspondingly very simple and includes the message types (type maps to a unique MooMS topic) to register interest for and the name of the Moolet to pass them to. For example, if you were integrating with the ACME system, which is an incident management system, you may have configuration as follows:

```
name : "ACMEIncidentManager",
classname : "CEmptyMoolet",
run_on_startup : true,
mooobot : "ACMEIntegration.js",
```

```

event_handlers : [
    "NewThreadEntry",
    "SigStatus"
]

```

- Be aware, not all event handlers are required for every integration, and you should only specify those handlers that are needed

AlertMgr

The out of the box Moog_farmd.conf file includes an example implementation of the Empty Moolet functionality in the form of the Alert Manager Moolet. We also ship Moobot for this Moolet named AlertMgr.js. An example use case for this Moolet is to enable some action on different Alert /Event types. For example, to update a Situation's services when an Alert is updated which contains certain attributes.

This Moolet requires the Alert to be passed directly to it via the Maintenance Manager Window or the Alert Builder Moolet using the process_output_of mechanism. Alternatively the standalone_moolet or mooms_event_handler mechanisms could be used to trigger Alert Manager behaviours.

Invoking custom functionality

To invoke custom javascript for a particular set of actions related to Situations you can leverage the Empty Moolet to listen for these actions and use the data within the Situations involved. For example, when a Situation is closed you may want to notify an external entity via the REST.V2 module.

Create Moobot

1. Edit moog_farmd.conf to associate a Moobot (CustomTaskRunner) with the EmptyMoolet, and listen for SigAction events:

```

{
    name          : "CustomTaskRunner",
    classname     : "CEmptyMoolet",
    run_on_startUp : true
    metric_path_moolet : false,
    moobot        : "CustomTaskRunner.js",
    event_handlers : [
        "SigAction"
    ]
}

```

Example of Moobot code that runs a function when a supported Situation action occurs in Cisco Crosswork Situation Manager:

CustomTaskRunner.js

```

var events = MooBot.loadModule('Events'); var logger =
MooBot.loadModule('Logger');
var constants = MooBot.loadModule('Constants');

logger.debug("Empty Moolet Started.");

/**
 * ### situationAction
 *
 * Listen for specific "sigAction"

```

```

*
* @param {object} situation - A situation object from Events
*/

```

```

function situationAction(situation) { logger.warning("Checking
Action event...");

var sitn_id = situation.value("situation_id");
var action = situation.payload().valueOf("action");

if (action !== null) {
    var details = situation.getActionDetails();
    // The name of the URL Tool has to match to trigger action if (action == "Ran Tool") {
        if (details.tool == urlToolName) {
            runFunction(sitn_id);
        }
    }
}

/***
* ### runFunction
*
* Run some function
*
* @param {number} sitn_id - The Situation Id
*/

```

```

function runFunction(sitn_id) {
    logger.info('Run some function for Situation Id ' + sitn_id);
}

//
// Listen for SigAction event to see if certain URL tool has been run
//

```

```
events.onEvent("situationAction",constants.eventType("SigAction")).listen();
```

The urlToolName must match the name of the Situation URL Tool. The Situation Id is available in the Event payload, because the tool is run in the context of a particular situation.

Situation Actions

"Created By Merge"

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

original_situations		list of Situation Ids that were involved in the merge to make a new Situation

Enricher Moolet

Enable the Enricher Moolet to update alerts with Enrichment data. See [Enrichment](#) for further information.

Configure Enricher

Define the behavior for the Enricher Moolet in the "Enricher" block of \$MOOGSOFT_HOME/config/moog_farmd.conf.

Enricher Parameters

The parameters that relate to the Enricher [Moolet](#) are as follows:

run_on_startup

Determines whether Enricher runs when Cisco Crosswork Situation Manager starts. If enabled, Enricher updates alerts with enrichment data from the moment the system starts, without you having to configure or start it manually.

Type: Boolean

Default: false

persist_state

Enables Enricher to save its state for [High Availability](#) systems so if a failover occurs, the second moogfarmd can continue from the same point.

Type: Boolean

Default: false

metric_path_moolet

Determines whether Enricher is factored into the [Event Processing](#) metric for [Self Monitoring](#) or not.

Type: Boolean

Default: false

description

Describes the Moolet.

Type: String

Default: Alert Enrichment

The default Enricher parameters are as follows:

```
{  
    name : "Enricher",  
    classname : "com.moogsoft.farmd.moolet.enricher.CEnricherMgr",  
    run_on_startup : true,  
    persist_state : false  
    metric_path_moolet : true,  
    process_output_of : "AlertBuilder", description : "Alert Enrichment"  
}
```

name and classname are hardcoded and should not be changed.

Output Parameters

These parameters control the output processed by the Moolet:

process_output_of

Defines the source of the alerts that Enricher processes. By default, the Moolet connects directly to the Alert Builder.

Type: List
One of: [AlertBuilder](#), [AlertRulesEngine](#)
Default: AlertBuilder

Housekeeper Moolet

The Housekeeper Moolet performs the periodic background tasks required for the Auto Close feature and for the moving of data to the historic database. Refer to the [Historic Database](#) document and the [Historic Data Utility Command Reference](#) for information on configuring the retention of historic data.

The Housekeeper Moolet is also responsible for gathering statistics from the system, for example Team Insights.

To verify the Housekeeper Moolet is running, use the following command:

```
ha_ctl -v
```

Configure Housekeeper

Define the behavior for the Housekeeper Moolet in the "Housekeeper" block of \$MOOGSOFT_HOME/config/moog_farmd.conf.

Housekeeper Parameters

run_on_startup

Determines whether Housekeeper runs when Cisco Crosswork Situation Manager starts. If enabled, Housekeeper performs its background tasks from the moment the system starts, without you having to configure or start it manually.

Type: Boolean
Default: true

persist_state

Enables Housekeeper to save its state for [High Availability](#) systems so if a failover occurs, the second moogfarmd can continue from the same point.

Type: Boolean
Default: false

metric_path_moolet

Determines whether Housekeeper is factored into the [Event Processing](#) metric for [Self Monitoring](#) or not.

Type: Boolean
Default: false

standalone_moolet

Determines whether the Housekeeper can listen for events generated by other Moolets within the same moogfarmd instance without being in a processing chain.

Type: Boolean
Default: true

The default Housekeeper parameters are as follows:

```
{  
    name          : "Housekeeper",  
    classname     : "com.moogsoft.farmd.moolet.housekeener.  
    -----"
```

```

run_on_startup      : true,
persist_state       : false,
metric_path_moolet: false,
standalone_moolet :

```

name and classname are hardcoded and should not be changed.

Maintenance Manager Moolet

Introduction

The Maintenance Manager Moolet compares alerts sent from the AlertBuilder moolet against active Maintenance Windows (see [Maintenance Schedule](#)) and then filters them.

If the alerts match an active Maintenance Schedule filter then they are not forwarded onto the next part of the chain, usually to a Sigaliser Moolet to be clustered into a Situation.

Please note: Maintenance Schedule is used when you have scheduled outages and do not want new Situations to be created. The Maintenance Manager Moolet ensures alerts will not be passed along to Sigalisers and clustered into Situations during that time period.

Configuration

The functionality of the Maintenance Window Manager is controlled by a Moolet in moog_farmd which has the default configuration shown below:

```
{
  name          : "MaintenanceWindowManager",
  classname     : "CMaintenance",
  run_on_startup: true,
  persist_state : false,
  metric_path_moolet: true,
  maintenance_status_field : "maintenance_status",
  maintenance_status_label : "In maintenance",
  update_captured_alerts : true
}
```

The following fields can be configured to change the behaviour of the Maintenance Manager Moolet:

Field	Input	Description
name	String	<p>The name of the Moolet. This should not be changed</p> <p>Please note: Only use the default 'MaintenanceWindowManager' as the name</p>
classname	String	The class name 'CMaintenance' is hardcoded and cannot be changed
run_on_startup	Boolean	If enabled, the Moolet will run when moogfarmd is started

<code>persist_state</code>	Boolean	If enabled, persistence will be turned on (state will be persisted in a cluster)
<code>metric_path_moolet</code>	Boolean	If enabled, the Moolet will be included in the "event_processing_metric" calculation in Self Monitoring
<code>process_output_of</code>	<code>AlertBuilder</code> <code>AlertRulesEngine</code> <code>Enricher</code>	This tells the Moolet to process the output of the Alert Builder, Alert Rules Engine or Enricher
<code>maintenance_status_field</code>	<code>String</code>	The name of the <code>custom_info</code> field/key used to indicate maintenance status
<code>maintenance_status_label</code>	<code>String</code>	The value of the <code>custom_info</code> maintenance status field used to indicate that an alert is in maintenance
<code>update_captured_alerts</code>	<code>Boolean</code>	<p>If enabled, ensures the maintenance status of an alert is set to null once the Maintenance Window that captured it has expired</p> <p>If disabled, the maintenance status field of a captured alert will remain as "In maintenance" (or whatever the <code>maintenance_status_label</code> text is set to) until that alert reoccurs at which point all <code>custom_info</code> maintenance fields will be set to null</p>

It is possible to add a column in the alert view displaying the 'Maintenance Status' for each alert and the text visible in this column can be controlled by editing the `MaintenanceWindowManager` Moolet configuration within `$MOOGSOFT_HOME/config/moog_farmd.conf` (edit the `maintenance_status_label`).

For the feature to function, the `MaintenanceWindowManager` Moolet needs to be placed before a Sigalising Moolet in a forwarding chain (configured in `$MOOGSOFT_HOME/config/moog_farmd.conf`). It is also appropriate to locate it before the `AlertRulesEngine` in the processing chain. This is the clean install configuration.

Updating Catured Alerts

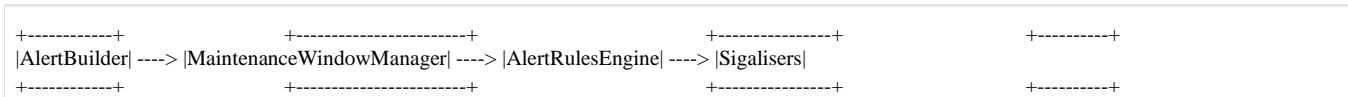
In addition to implementing the maintenance windows, the `MaintenanceWindowManager` Moolet also updates each alert (affected by a maintenance window) with several `custom_info` fields.

By default an alert that is "captured" by a maintenance window will have the following fields set:

Field	Description
<code>custom_info.maintenance_status</code>	Configurable text label - set to "In maintenance" by default
<code>custom_info.maintenance_id</code>	The numerical id of the maintenance window that captured the alert
<code>custom_info.maintenance_name</code>	The name of the maintenance window that captured the alert
<code>custom_info.forward_Alerts</code>	Whether the alert is forward to Sigalisers or not - false by default

Out-of-the-Box Moolet Flow

The default, or out-of-the-box, flow for the Moolet looks something like this:



To allow programmatic forwarding of alerts in the Moobots to different Sigalisers from an `AlertBuilder` (using `alert.forward('SigaliserName');`), you need more than one 'Maintenance Window Manager' Moolet. Here is an [example](#) of a selection of different approaches within a single `moogfarmd` instance:

- AlertBuilder1 -> if (x) then alert.forward('MaintenanceWindowManager1'); -> MaintenanceWindowManager1 -> Sigaliser (process_output_of 'MaintenanceWindowManager1')
- AlertBuilder1 -> if (y) then alert.forward('MaintenanceWindowManager2'); -> MaintenanceWindowManager2 -> Cookbook1 (process_output_of 'MaintenanceWindowManager2')
- AlertBuilder2 -> alert.forward('MaintenanceWindowManager3'); -> MaintenanceWindowManager3 -> Cookbook2 (process_output_of 'MaintenanceWindowManager3')
- AlertBuilder3 -> alert.forward(this); -> MaintenanceWindowManager4 -> Speedbird (process_output_of 'MaintenanceWindowManager4')

There are also endpoints available to manage this feature with the Graze API.

Create new maintenance window (HTTP POST):

To create a **one-time** maintenance window (no recurrence), which is filtered on 'source equal to "hostWhichIsDown":

```
curl "https://<YOUR_HOSTNAME>:8080/graze/v1/createMaintenanceWindow" -H "Content-Type: application/json; charset=UTF-8" --insecure -X POST -v -- data '{"auth_token": "<YOUR_GRAZE_AUTH_TOKEN>", "name": "my_window_1", "description": "This is my description", "filter": { "column": "source", "op": 0, "value": "hostWhichIsDown1", "type": "LEAF" }, "start_date_time": 1473849237, "duration": 55800, "forward_alerts": false}'
```

To create a maintenance window (same filter as above) that recurs once a **month** (from its start_date_time), add the recurring_period (only allowed value is 1) and recurring_period_units properties (allowed values are 2 (daily), 3 (weekly) and 4 (monthly)):

```
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": "hostWhichIsDown1", "type": "LEAF" }, "start_date_time": 1473849237, "duration": 55800, "forward_alerts": false, "recurring_period": 1, "recurring_period_units": 4}'
```

Delete maintenance window (HTTP POST):

```
curl "https://<YOUR_HOSTNAME>:8080/graze/v1/deleteMaintenanceWindow" -H "Content-Type: application/json; charset=UTF-8" --insecure -X POST -v -- data '{"auth_token": "<YOUR_GRAZE_AUTH_TOKEN>", "id":123}'
```

Get current or future maintenance windows

The selection returned can be controlled using the **start** and **limit** parameters) (HTTP GET):

```
curl "https://<YOUR_HOSTNAME>:8080/graze/v1/getMaintenanceWindows?auth_token=<YOUR_GRAZE_AUTH_TOKEN>&start=0&limit=20"
```

- This will not return deleted Maintenance windows or Maintenance windows which have expired/set in the past

Important Note on Maintenance Windows via Graze

The following limitations are in effect:

- The "filter" syntax must be in the correct Moog JSON format (same format as used in alert and Situation filters in the DB)
- The "start_date_time" property must be in epoch time
- For a recurring maintenance window, the recurring_period property must be 1 - no other value will be accepted

- For a recurring maintenance window, the recurring_period_units property has allowed values of 2 (daily), 3 (weekly) or 4 (monthly) - no other values will be accepted

This feature uses custom_info fields within the alerts. As a result, Moobots should not overwrite the following fields or completely empty the custom_info object within alerts:

- custom_info.maintenance_status
- custom_info.maintenance_id
- custom_info.forward_alerts

Notifier Moolet

Introduction

The Notifier Moolet enables Cisco Crosswork Situation Manager to act on inviteMooMS Bus messages and optionally send an email.

For example, to send an email when a user is invited to a Situation via the UI, the Notifier Moolet must:

- Listen to Invite Events
- Interrogate the Events to identify Situation invitations
- Filter out Events for Situations we are not interested in notifying
- Extract relevant information from the Event including the Situation Id and User Id
- Send an email message containing a customized body to a recipient using the Mailer Moobot module

Moogfarmd Configuration

The Notifier Moolet is designed to take messages off the MooMS bus according to message type. The configuration is as follows:

```
{
    name          : "Notifier",
    classname     : "CNotifier",
    run_on_startup: false,
    metric_path_mookelet : false, moobot      :
    "Notifier.js"
}
```

Scheduler Moolet

- [Introduction](#)
- [Scheduling Frequency](#)
- [Constraints](#)

Introduction

The Scheduler Moobot runs scheduled jobs at regular intervals throughout the lifetime of moog_farmd.

```
moog_farmd.conf:

# The Scheduler is used to run scheduled jobs at regular
# intervals throughout the lifetime of moog_farmd. Only this # moolet, which cannot
# subscribe to the MooMS bus and
# listen to events, is allowed to submit scheduled jobs. #
# To start up successfully it must have the name and threads
# values set to "Scheduler" and 1 respectively.
```

```
{
    name                  : "Scheduler",
    classname             : "CScheduler",
    run_on_startup        : false,
    persist_state          : false,
    metric_path_moolet    : false,
    moobot                : "Scheduling.js",
}

}
```

To load the scheduler module:

```
var scheduler =MooBot.loadModule('Scheduler');
```

Run jobs using, for example:

```
// A job that fails and does not restart. scheduler.scheduleJob(this, "knockOnce",
5, 5, false);
```

This calls a method in the same js file called knockOnce:

```
function knockOnce()
{
    logger.warning("Knock knock");
    throw new Error("Failed to knock.");
}
```

The scheduledJob method has two possible parameter sets:

- `scheduleJob(this, functionName, start_delay , interval);`
- `scheduleJob(this, functionName, start_delay, interval, true | false) ;`

Parameter	Description
first	always this
second	the name of the function to call to run the job
third	is the delay from starting farmd to the first run (in seconds)
fourth	the interval between runs (in seconds)
fifth	decides whether the job will run again if it failed previously

Scheduling Frequency

When executing multiple jobs we recommend that you try and offset potential workload, by for example staggering the initial run of multiple jobs a few seconds apart or scheduling jobs at slightly offset frequencies.

Constraints

1. Must be single threaded.
2. Only one per moog_farmd process.
3. Has to be called Scheduler.
4. Use a Moobot module function as a scheduled job - which involves some rarer JavaScript as the scheduler won't recognise function names from modules (it can't find them)

For example, in your scheduler moobot you might have:

```

MooBot.loadModule('AutoClose.js');
var autoClose=new AutoClose();

// Bind the module function locally to the module function.

var autoCloseAlertFunction = autoClose.closeAgedAlerts.bind
(autoClose);

// Schedule execution

scheduler.scheduleJob(this, "autoCloseAlertFunction" , 60,
autoCloseAlertFrequency . true);

```

Situation Manager

- Configure the Situation Manager
- Moobot Configuration

Situation Manager listens for Situations being created, updated or closed and passes them to an associated Moobot.

You can determine the origin of the Situations that are processed, which Moobots the Situations are passed to and what functions occur to them.

When the Moobot receives a Situation, it can be configured to perform functions such as data enrichment, auto-assignment to a user or notifying third-party tool to raise a ticket.

Configure the Situation Manager

You can configure the Situation Manager in the moog_farmd.conf file.

The Situation Manager parameters that can be configured are as follows:

run_on_startup

Determines whether Situation Manager runs when Cisco Crosswork Situation Manager is started or not.

Type: Boolean
Default: false

metric_path_moobot

Determines whether Situation Manager is factored into the Event Processing metric for Self Monitoring or not.

Type: Boolean
Default: false

moobot

Specifies a JavaScript file found in \$MOOGSOFT_HOME/bots/moobots. Examples of available Moobots include SituationMgr.js, SituationMgrLabeller.js or SituationMgrNetcool.js. These can be customized to meet your needs.

Type: String
Default: "SituationMgr.js"

mooms_event_handler

When set to true, listens to messages sent on the sigs topic by any moobleet running on a different farmd instance. This option directly replaces standalone, which is now deprecated but preserved for backwards-compatibility.

Type: Boolean
Default: true

process_output_of

Determines which algorithms Situation Manager processes the output of. This might be a single Sigaliser algorithm or multiple algorithms.

Type: List
Default: "Sigaliser", "TemplateMatcher", "Speedbird", "Default Cookbook".

An example Situation Manager configuration is as follows:

```
{  
    name: "SituationMgr", classname:  
        "CSituationMgr", run_on_startup: true,  
        metric_path_moolet: true, moobot:  
            "SituationMgr.js",  
            mooms_event_handler: true,  
            process_output_of: [  
                "Sigaliser_1",  
                "Sigaliser_2",  
                "Sigaliser_3",  
                "TemplateMatcher",  
                "Speedbird", "Cookbook"  
            ]  
}
```

Please note: The classname is hardcoded and should not be changed.

Moobot Configuration

Situation Manager listens for three event types by default:

- Sig - a new Situation being created. SigClose -
- a Situation being closed. SigUpdate- a
- Situation being updated.

If you want to listen for other events, create an Empty Moolet and define the other events under event_handler. The full list of event types refer to the [Constants.eventType](#) section.

You can listen for specific Situation actions using the SigActionevent. It can filter out the following actions:

Assigned Moderator Situation Resolved Situation Revived Situation Closed Assigned Queue Created By Merge Used In Merge Created By Split Used For Split Ran Tool Acknowledged Situation Moderator	Deacknowledged Situation Moderator Added Alerts To Situation Added Entry To Thread Changed Situation Processes Changed Situation Services Created Thread Agreed With Thread Entry Commented On Thread Entry Disagreed With Thread Entry Changed Situation Custom Info	Described Situation Excluded User Invited User Moved Alerts To Situation Removed Alerts From Situation Situation Teams Changes Marked Thread Entry As Resolving Unmarked Thread Entry As Resolving Situation Rating Situation Rating Removed
--	---	--

There are three Moobots available in \$MOOGSOFT_HOME/bots/moobotsthat Situations can be sent to by Situation Manager. These can be customized to meet your requirements:

SituationManager.js

Situation Manager's default associated Moobot is called SituationManager.js.

For more information on how it can be configured see [Moobots](#).

SituationMgrLabeller.js

Another application of the Situation Manager is Situation Manager Labeller.

This can be used to enrich Situations by dynamically adding Alert properties to the Situation description.

SituationMgrNetcool.js

This Situation Manager moobot is required for the Legacy Netcool integration.

For more information see [Legacy Integration for Netcool](#).

Speedbird Moolet

- Speedbird's Algorithm
- Configuration
- Tuning guidelines

Speedbird groups events related to an actionable outage into clusters of their related Alerts. These clusters are service impacting, with the group of 'clustered' Alerts providing operational value to someone using the system.

Speedbird allows you to configure a set of parameters of an Event to drive the clustering in addition to time. For example, you may want a group of Alerts and Events together that have a co-incidence in time, but also have a coincidence in another value of the Event, such as, the hostname. Speedbird allows you to create clusters of Alerts with a similar hostname that have also occurred at a similar time.

Speedbird's Algorithm

The algorithmic technique used by SpeedBird is based around K-means, which is a well-understood and traditional clustering algorithm that is a form of unsupervised machine learning. For the SpeedBird Moolet, Cisco Crosswork Situation Manager uses some of the same algorithmic tool chain that is used in the Sigaliser along with the K-means algorithm. For instance, Cisco Crosswork Situation Manager still uses the same time based determination of how many real clusters there are in the data at a given point in time. Non-negative matrix factorisation in the limit collapses into a K-means calculation, but is more computationally efficient.

Configuration

To configure SpeedBird, the following should be read in conjunction with the [Tuning guidelines](#), to enable you to produce optimal results.

sig_resolution

In **moog_farmd.conf**, there is a general **sig_resolution** parameter grouping before the Moolet definitions with the following parameters:

```
sig_resolution :  
{  
    alert_threshold : 1,  
    sig_similarity_limit : 0.7  
},
```

- These parameters are set for all Sigalisers whether it is SpeedBird or the traditional Sigaliser running in a given farmd. The **sig_resolution** parameters allow you to compare pre-existing Situations and determine if it is an evolution of an existing Situation, or, a new Situation

Moolet and Algorithm

The parameter groups Moolet and Algorithm function in the same way as those in the existing Sigaliser.

```
# Moolet  
name : "Speedbird", classname :  
"CSpeedbird", run_on_startup : false,  
process_output_of : "AlertBuilder",
```

```
# Algorithm time_compression : true,
scale_by_severity : true,
entropy_threshold : 0.35,
```

For further information on these parameters, see the table below:

Parameter	Input	Description	Example
name	-	The name of the Sigaliser	Speedbird
classname	-	The classname of the Sigaliser This is hardcoded and should never be changed	CSpeedbird
run_on_startup	Boolean	If enabled, an instance of the Sigaliser will be created when moog_farmd starts. This is disabled by default	false
process_output_of	AlertBuilder AlertRulesEngine	This sets whether the Sigaliser processes the output of either the Alert Builder or Alert Rules Engine. The latter can only be used if automations are desired prior to the Situation resolution Please note: The Sigaliser can only have one input	AlertBuilder
time_compression	Boolean	If enabled the Sigaliser will ignore empty time buckets. If disabled, it will include empty timebuckets Please note: For low data rates you should set this to 'true', for normal data rates set this to 'false'	true
scale_by_severity	Boolean	If enabled, high severity Alerts are treated as having higher entropy. This scaling is done as the severity constant number divided by the maximum severity (5)	true
entropy_threshold	Integer	The value of this parameter is the minimum entropy that the Alert must have to be included in the Sigaliser calculation. Any Alert that arrives at the Sigaliser with a lower entropy than this value will not be included in Situations. Please note: The default value of 0.0 means every Alert will be processed by the Sigaliser	0.35

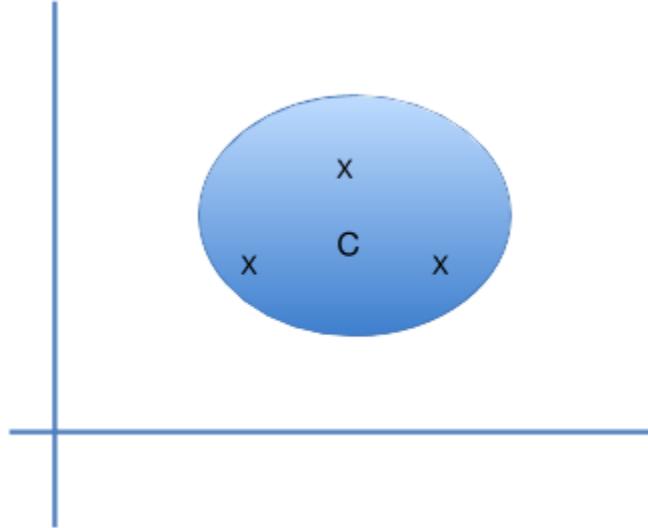
sig_alert_horizon

The `sig_alert_horizon` parameter allows you to prune clusters. The value allows you to control when you remove outlying Events from the cluster:

- If the value is less than <0.0, no pruning is undertaken.
- At 0.0, members that are further than one standard deviation from the centroid of the cluster are eliminated.
- At more than > 0.0 the standard deviation is multiplied by `sig_alert_horizon`, and then members further than `mean + sig_alert_h orizon*std_dev` distance from centroid are eliminated.

```
sig_alert_horizon : 0.0,
```

Every cluster has a centroid, which is the average point in the middle of a cluster.



In the diagram above there are three points in a defined cluster (X), and the centroid (C), which is not a real point in this space phase but represents the center of the cluster. You compute the distance of each point from the centroid of the cluster, which results in an average distance and standard deviation. You can then work out the standard deviation to determine the spread of the cluster. A low standard deviation, i.e., 0, means all of the points are the same distance from the centroid; whereas, a high standard deviation means they are a highly variable distance from the centroid thus indicating a random cluster.

components

You can choose which parameters of an Event are used by the clustering algorithm. In the following example, "source", "source_id", "description" are declared:

```
components : [ "source","source_id","description" ]
```

Additionally, the system always takes into consideration the time that the Event arrives in the system (`event_time` or `last_occurred` for an Alert). You can have as many components as you like, but, the more components that are selected, the greater numerical complexity is introduced into the system, and there is a chance you will get a smaller number of Alerts per cluster and less correlation.

Partitioning

There are two methods of partitioning the data into Situations. The first is '`partition_by`' which splits the clusters according to the parameters specified. The second is '`pre_partition`', which splits the incoming Event stream before clustering.

Please note: Pre-partitioning is recommended as it does not interfere with the results of the clustering algorithms

`partition_by`

After clustering has taken place and before you enter merging and resolution, you can split clusters into sub-clusters based on a component of the Events. For example, you can use the managerparameter to ensure the Situations only contain Events from the same manager. In general, and by default, you should comment out the partition_byparameter.

pre_partition

An alternative way of partitioning is to use pre_partition which allows you to specify a component field (from the list of specified components) around which the Event stream will be partitioned before the K-means clustering occurs. The Alerts in the resulting Situations will each contain a single value for the component field chosen.

For example, if the SpeedBird componentoption was set to:

```
components : [ "source","manager","description" ],
```

In the metricbelow, the description component is being weighted more heavily compared to source and manager. Please note that the metric always contains one more values than the components specified and that the first value always corresponds to time.

```
default : [ 1,1,1,1000000],
```

This results in Situations containing Alerts with more similar descriptionfields and a variety of sourceand managerfields.

Adding the following property ensures that Situations contain Alerts with very similar descriptionfields, a variety of sourcefields but only a single distinct managerfield.

```
pre-partition : "manager"
```

pre_partition, like partition_by, is defaulted to false in **moog_farmd.conf** so has no effect. If pre_partitionis not required there is no need to modify the existing **moog_farmd.conf** files to include the property.

It is possible to configure pre_partitionand partition_byat the same time, but the partition_byparameter will only have any effect if it is applied to a different component.

A note on time_compression and pre_partition

pre_partitionsplits the Events into separate streams based on the component you have specified, as opposed to partition_by, which allows the algorithms to work on the whole Event stream and then splits up the results.

Partitioning the Event stream using pre_partition can make time_compression less effective. There are many things in the tuning parameters and behaviours of the Sigalisers that depend upon the event rate, and because you are splitting the stream up, if you have an event rate of X and you split it into many streams, each of those streams is going to have an event rate of less than X. This can skew whether the tuning parameters you are using are appropriate, so with or without time_compression you should be careful. With time_compression, you expect to avoid silent moments in the Event stream, but this may not be the case because the effect of pre_partitionis to split the stream.

For example, if you pre_partitionon manager, set time_compressionto true, and set windowto 10and resolution to 60, you will store up to 10 one-minute wide buckets of Events for clustering.

The Events could arrive as follows:

Bucket	Minute	Manager
1	1	Andrew, Alan
2	2	Alan
3	17	Alan
4	18	Alan
5	20	Andrew
6	35	Alan
7	37	Alan

8	38	Alan
9	57	Alan
10	59	Alan
11	60	Alan

It should be noted that the minute 1 bucket will be dropped from the Sigaliser window because Cisco Crosswork Situation Manager only keeps the last ten live buckets. Clustering for Events with Manager Alan will only use nine buckets, and clustering for Events with Manager Andrew will only use 1 bucket.

metric

```
metric : {
    default : [ 1,1,1,1], categoryField:
    "agent",
    "DBMON" : [ 100,1,1000,1000000],
    "NETMON" : [ 1,100000000,1,0]
```

The metric is a technical and detailed area of configuration, which relates to how Cisco Crosswork Situation Manager measures distance between two events in the phase space used for clustering. Euclidean distance is easy to compute as you calculate the square of the differences in the components (in two dimensions the distance is the hypotenuse of a right-angled triangle, in three dimensions it is the diagonal measurement of a cuboid, and so on...) add them all up and this reveals the square of the distance. This example is a simplification.

For instance, if you have x, y and z as the components of a vector, the square root of the distance is:

$$A = \{x_1, y_1, z_1\}, B = \{x_2, y_2, z_2\}$$

$$d^2 = a_1(x_2 - x_1)^2 + a_2(y_2 - y_1)^2 + a_3(z_2 - z_1)^2$$

You can put a number in front of these sums of squares, and the values are more correctly known as the diagonal metric tensor values. Cisco Crosswork Situation Manager assumes that you should only ever consider the diagonal metric tensor values; however, in general co-ordinate geometry you can contribute to the distance by adding in, for example, $(y-z)^2$. It is not considered useful to compare different attributes of an event for similarity.

This approach allows you to weight the distance between two events based upon their components. For example, if X represents time, Y represents source and Z represents manager, and you make a2 much bigger than a1. Any distance in source creates a lot more distance between the events than the same distance in time. This allows you to weight the importance. This is why you have four component values in all the different metrics. The default is [1,1,1,1]. You can also select a category Field, which is a parameter in the event, i.e., categoryField: "agent".

In the example configuration above, if one of the events has a value DBMON, then you use the metric [100,1,1000,1000000] to weight the distance; otherwise, if NETMON, you use the alternate metric [1,100000000,1,0]. If you have neither of these two values, you use default. This allows configuration of different metric weightings for different sources of events.

string_len_cutoff

This determines the maximum number of characters in a component to use in the distance calculation described in the previous section. This cutoff will apply to all string components being used.

For example, if there are occasionally very long descriptions, you can specify a 64-character cutoff which will avoid excessive computation. See example below:

```
string_len_cutoff : 64
```

spread_cutoff

Whereas the sig_alert_horizonis used to take events out of clusters, spread_cutoffdetermines whether or not to consider a cluster to be worth processing.

```
spread_cutoff : 5.0
```

- 0 means all clusters have to be one hundred percent tight, so the same distance from the center with no variation; otherwise, the cluster will be discarded. A higher number allows for looser clusters, i.e. more variation within the cluster.

The spread cutoff uses the cluster standard deviation, after any outliers have been pruned in accordance with the `sig_alert_horizonparameter` to determine, which clusters should be rejected. 0.0 means that all clusters have to be one hundred percent tight, i.e., with all members matching the cluster centroid. A higher number allows for more loosely correlated clusters. It is worth noting that the metrics chosen for weighting the components can have a direct impact on the standard deviation of the clusters generated, and it may be necessary to increase the `spread_cutoff` value to reflect this.

ignore_case

When comparing strings, determines if the translation of strings into a number in 'phase' space is case sensitive. In general, case should be ignored. See below:

```
ignore_case : true,
```

iterations

Unless "Entropy" seeding is specified, the initial seeds for K-means clustering includes a random element that will lead to different solutions on different iterations. If more than one iteration is chosen Speedbird will select the best solution of those returned for Situation processing. For higher numbers of iterations, K-means clustering will tend to converge on an optimal solution, which in turn leads to lower variance from one Speedbird run to another. Iterations however take both time and CPU resources so a sensible compromise between speed and the optimal solution is needed.

```
iterations : 5,
```

- Cisco recommends a value of 5

seeding

Seeding can be set to 'Kmpp', 'Lloyd', or 'Entropy'. Both 'Kmpp' (recommended) and 'Lloyd' use random elements to select seeds to initialise the clustering process, and therefore have the advantage of finding different cluster solutions over multiple iterations.

Alternatively, 'Entropy' selects the highest entropy Events to seed clusters, and as such, returns the same results on each occasion. It should be noted that this is not necessarily an optimal result.

```
seeding : "Kmpp",
```

force_causal

Setting `force_causal` to true ensures that Events which are part of causal Alerts are preserved. They are never discarded during the K- means clustering process, but are always returned as a member of a cluster.

The entropy range for causal alerts is defined in the `moogdb.significantable`.

```
force_causal : true
```

generate_stats

`generate_stats` provides detailed logging useful for tuning purposes. Detailed logging is written at log level `WARN` to the **moogfarmd.log** file. The logging contains detailed information around event clustering, and also includes information about partitioning.

```
generate_stats : "true"
```

- If generate_stats is not required there is no need to modify an existing **moog_farmd.conf** files to include the property

Tuning guidelines

To ensure you produce useful results, it is recommended that you read the following in conjunction with description of the configuration parameters:

- 1 Disable parameters which remove Alerts from Situations and discard Situations which are poorly correlated. Start with `sig_alert_hori_zon` set to -0.1 (to prevent any outliers from being pruned) and `spread_cutoff` set to a high value (to prevent any clusters from being discarded). Subsequently modify these parameters to reduce Situation size and numbers.
- 2 When tuning the system, consider using 'Entropy' seeding and only switching to 'Kmpp' when you are happy with the results. 'Entropy' seeding always produces the same Situations, unlike 'Kmpp' or 'Lloyd', but often not the most appropriate ones. Using 'Entropy' seeding guarantees you can normally run a dataset once to see if the parameters you have used have given you the desired effect. 'Kmpp' seeding usually produces the best Situations with a moderate number of iterations.
- 3 The K in K-means indicates the number of seeds Cisco Crosswork Situation Manager clusters around and the number of Situations which are produced. It is calculated using a technique that analyses the dataset to establish the number of independent clusters of events. The calculation is dependent on number of time slices (window), and the effective event rate (after entropy thresholds etc.) which determines the number of unique signatures received in resolution*window seconds.

Please note: Moogsoft advises that you start by asking how many tickets/Situations are expected in a day, and you adjust the resolution/windowparameters to achieve the same number of Situations in a day's worth of data. The value of k is never greater than the windowand the number of unique alerts in the total window, and is often about 80% of this value

- 4 If you are using time and one other component, be prepared to significantly reduce the time metric, as well as, increasing the value of the metric for the other component. For example, assume that you have the following configuration and that you are interested in a series of events that occur over 10 minutes:

```
components : ["source"] metric : {
    default : [1,1000000]
}
```

The time spread of the cluster you are interested is 600 seconds. If you have increased the metric a lot on the source component, your cluster may contain a single value for source (or very closely related values). Therefore, the cluster spread value will be generated largely or entirely by the event time component. K-means solutions that split this set of events into more than one cluster are preferred over those that keep them in a single cluster. If you use default metrics [1,1], the clustering will mostly be primarily driven by time.

- 5 The metrics that you use may affect the `spread_cutoff`. If you increase a metric it may be necessary to increase the spread cut-off by quite a large amount (up to the square root of the increase of the metric).
- 6 It is the square root of the metric that is applied to a component. If you increase a metric for a component from 1 to 100, you emphasise the effect of that component on the resulting clusters by a factor of 10.
- 7 Do not vary more than one configuration parameter at a time.
- 8 Start with small data sets and limited (i.e., time plus one other) components before increasing the size and, or, complexity of your solution.

Template Matcher Mooler

Template Matcher was deprecated from the release of Cisco Crosswork Situation Manager V7.0.0. For more information see [Deprecation Notice: Template Matcher](#).

The Situation Templates enable users to provide feedback to Cisco Crosswork Situation Manager that indicates the quality of the Situations produced and consequently what Cisco Crosswork Situation Manager should do if it detects a similar situation in the future.

moog_farmd

The TemplateMatcher moollet should be activated in moog_farmd if you wish your system to discover "Priority Templates". By default, moog_farmd will always suppress the creation of SPAM templates, regardless of the moollet they are generated by.

There is no special configuration required for the moollet; it is either "on" or "off". TemplateMatcher should run after the AlertBuilder or AlertRulesEngine depending upon the system configuration. The default configuration for the TemplateMatcher is shown below:

```
{  
    name : "TemplateMatcher",  
    classname : "CTemplateMatcher",  
    ...  
    ...  
    #process_output_of : "AlertRulesEngine",  
    process_output_of : "AlertBuilder", threads : 4  
}
```

Standalone Execution

The TemplateMatcher moollet can be executed in a "stand-alone" mode via the moog_primer/signaliser utility using the following command:

```
signaliser --moollet=TemplateMatcher
```

Teams Manager Moollet

The Teams Manager Moollet is triggered by Cisco Crosswork Situation Manager when a Situation is created, updated and deleted, and also when a team is created and updated. You can assign teams to Situations using the filters in the UI under **Settings > Teams > General**. If there are no filters for a team, it is assigned all new Situations by default.

If you use the "assignTeamsToSituation" Graze API endpoint or MoogDb method to assign teams to a Situation, Cisco Crosswork Situation Manager marks the Situation as overridden. The Teams Manager Moollet can no longer act on it even if that Situation matches a filter.

You can alter the behavior of the Teams Manager Moollet by changing the "Situation Update Policy" in the UI under **Settings > Teams**.

One Teams Manager Moollet is run for every instance of Cisco Crosswork Situation Manager.

Configure Teams Manager

The Teams Manager Moollet configuration is defined in the "TeamsMgr" block of \$MOOGSOFT_HOME/config/moog_farmd.conf.

Teams Manager Properties

The properties that relate to the Teams Manager Moollet are:

run_on_startup: Determines whether Teams Manager runs when Cisco Crosswork Situation Manager starts. If you enable it, Teams Manager processes Moollet output from the moment the system starts, without you having to configure or start it manually.

Type: Boolean
Default: true

persist_state: Enables Team Manager to save its state for [High Availability](#) systems so if a failover occurs, the second moogfarmd can continue from the same point.

Type: Boolean
Default: false

metric_path_moollet: Determines whether Cisco Crosswork Situation Manager factor Teams Manager into the [Event Processing](#) metric for [Self Monitoring](#) or not.

Type: Boolean
Default: false

moobot: JavaScript program that controls and customizes the behavior of Teams Manager.

Type: String
Default: "TeamsMgr.js"

The default Teams Manager configuration is:

```

name : "TeamsMgr",
classname : "CTeamsMgr",
run_on_startup : true,
persist_state : false,
metric_path_moolet : false,
moobot

# Specifies the list of all the moolet that can change # or create situations.
Remove this section if the
# TeamsMgr is running in its own instance. #
process_output_of : [
    "Sigaliser",
    "TemplateMatcher",
    "Speedbird", "Cookbook",
    "Default Cookbook",
    "Nexus", "SituationMgr"
]

```

`name` and `classname` are hardcoded and should not be changed.

Output Parameters

These parameters control the output the Moolet processes:

`process_output_of`: The Moolets that perform actions that trigger the Teams Manager:

Type: Array

Valid Moolets: `Sigaliser`, `TemplateMatcher`, `Speedbird`, `Cookbook`, `Default Cookbook`, `Nexus`, `SituationMgr` **Default:** ["Sigaliser", "TemplateMatcher", "Speedbird", "Cookbook", "Default Cookbook", "Nexus", "SituationMgr"]

Graze API

- [Architecture](#)
- [Tomcat Configuration](#)
- [API Definition](#)
- [Authentication Troubleshooting](#)
- [Endpoints](#)
- [POST parameters](#)
- [HTTP status and error codes](#)

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

Architecture

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

Tomcat Configuration

Tomcat must be configured to allow cross-context calls to be made by adding the following to the context.xml file in the Tomcat \$APPERVER_HOME/conf directory:

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

API Definition

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

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

For example:

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

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

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

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

Authentication Troubleshooting

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

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

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

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

- The user is not assigned the Grazer role:

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

- No user of that name:

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

- Incorrect password:

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

Endpoints

Alerts

addAlertCustomInfo

addAlertCustomInfo

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

addAlertToSituation

addAlertToSituation

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

assignAndAcknowledgeAlert

assignAndAcknowledgeAlert

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

assignAlert

assignAlert

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

↳ [closeAlert](#)

closeAlert

A POST request that closes one or more alerts.

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

↳ [deassignAlert](#)

deassignAlert

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

Request Arguments

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

alert_id	Number	The alert ID
----------	--------	--------------

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

[getAlertDetails](#)

getAlertDetails

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

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

Name	Type	Description
active_sitn_list	Number list	A list of Situation IDs of the active Situations to which this alert belongs
agent	String	The agent name associated with this alert*
agent_location	String	The agent location associated with this alert*
alert_id	Number	The alert ID
class	String	The class associated with this alert*
count	Number	The number of times that this alert has occurred
custom_info	JSON Object	A JSON Object containing the custom information
description	String	The description associated with this alert*

entropy	Number	The entropy value of the alert, the measure of probability that an alert will arrive in the system at any given time. This is a value between 0 (very certain) and 1 (very uncertain).												
external_id	String	The external ID associated with this alert*												
first_event_time	Number	The timestamp (in Unix epoch time) of the first occurrence of this alert												
int_last_event_time	Number	The internal Cisco Crosswork Situation Manager timestamp (in Unix epoch time) of the last occurrence of this alert												
last_event_time	Number	The timestamp (in Unix epoch time) of the last occurrence of this alert												
last_state_change	Number	The timestamp (in Unix epoch time) of the last status change of this alert												
manager	String	The manager name associated with this alert*												
owner	Number	The User ID of the user that this alert is assigned to												
severity	Number	<p>The alert's severity as an integer:</p> <table border="1"> <tr><td>0</td><td>Clear</td></tr> <tr><td>1</td><td>Indeterminate</td></tr> <tr><td>2</td><td>Warning</td></tr> <tr><td>3</td><td>Minor</td></tr> <tr><td>4</td><td>Major</td></tr> <tr><td>5</td><td>Critical</td></tr> </table>	0	Clear	1	Indeterminate	2	Warning	3	Minor	4	Major	5	Critical
0	Clear													
1	Indeterminate													
2	Warning													
3	Minor													
4	Major													
5	Critical													
signature	String	The unique alert identifier												
significance	Number	<p>The alert's severity as an integer:</p> <table border="1"> <tr><td>0</td><td>Collateral</td></tr> <tr><td>1</td><td>Related</td></tr> <tr><td>2</td><td>Impacting</td></tr> <tr><td>3</td><td>Causal</td></tr> </table>	0	Collateral	1	Related	2	Impacting	3	Causal				
0	Collateral													
1	Related													
2	Impacting													
3	Causal													
source	String	The source associated with this alert*												
source_id	String	The source ID associated with this alert*												
state	Number	Indicates the lifecycle state of the alert												
type	String	The type associated with this alert*												

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

Example

Curl Command

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

Successful request return:

```
{  
    "active_sitn_list": [1], "agent": " TestBed ",  
    "agent_location": " localhost ", "alert_id": 10,  
    "class": " WebMon ",  
    "count": 2, "custom_info": null,  
    "description": " Web Server HTTPD is DOWN ",  
    "external_id": " 12345 ", "first_event_time": 1416307126,  
    "int_last_event_time": 1416307188,  
    "last_event_time": 1416307131,  
    "last_state_change": 1416307144, "manager": " WebMon ",  
    "owner": 2, "severity": 0,  
    "signature": " SIG:Web Server Down Trap:xldn1458pap:10 ", "significance": 3,  
    "source": " xldn1458pap ",  
    "source_id": " xldn1458pap ", "state": 9,  
    "type": " HTTPDDown "  
}
```

getAlertIds

getAlertIds

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

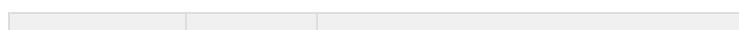
Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

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



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

Example

Curl Command:

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

Successful request return:

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

SQL-like Filters

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

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

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

removeAlertFromSituation

removeAlertFromSituation

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

[setAlertAcknowledgeState](#)

setAlertAcknowledgeState

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful return:

NO RESPONSE TEXT

[setAlertSeverity](#)

setAlertSeverity

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

Request Arguments

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

severity	Number	The alert's severity as an integer:
----------	--------	-------------------------------------

0	Clear
1	Indeterminate
2	Warning
3	Minor
4	Major
5	Critical

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful return:

NO RESPONSE TEXT

Situations

↳ [addSigCorrelationInfo](#)

addSigCorrelationInfo

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

```
[REDACTED]
```

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

Successful Return:

NO RESPONSE TEXT

[addSituationCustomInfo](#)

addSituationCustomInfo

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

Note: This method has superseded the now deprecated method addCustomInfo.

Request Arguments

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request
sitn_id	Number	The Situation ID
custom_info	JSON Object	A JSON Object containing the custom information

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

[addThreadEntry](#)

addThreadEntry

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

Threads are comments or 'story activity' on Situations.

Request Arguments

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

sitn_id	Number	The Situation ID
thread_name	String	The name of the existing thread
entry	String	The entry to add. For example, 'And another thing...'

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see HTTP status and error codes

assignAndAcknowledgeSituation

assignAndAcknowledgeSituation

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

assignSituation

assignSituation

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

assignTeamsToSituation

assignTeamsToSituation

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

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

Request Arguments

Include either team_ids or team_names.

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure. For codes, see below .

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

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

Example Curl Command 1:

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

Return:

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

Example Curl Command 2:

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

Return:

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

Unassign Example:

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

Return:

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

closeSituation

closeSituation

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

Request Arguments

Name	Type	Description						
auth_token	String	A valid auth_token returned from the authentication request						
sitn_id	Number	The Situation ID						
resolution	Number	Determines what to do with the Situation's alerts: <table border="1"><tr><td>0</td><td>Close no alerts</td></tr><tr><td>1</td><td>Close all alerts in this Situation</td></tr><tr><td>2</td><td>Close only alerts unique to this Situation</td></tr></table>	0	Close no alerts	1	Close all alerts in this Situation	2	Close only alerts unique to this Situation
0	Close no alerts							
1	Close all alerts in this Situation							
2	Close only alerts unique to this Situation							

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

createSituation

createSituation

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

- Status: Opened
- Moderator: none assigned

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object with the following:

Name	Type	Description
sitn_id	Number	The new Situation ID

Example

Curl Command:

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

Successful request return:

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

createThread

createThread

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

Threads are comments or 'story activity' on Situations.

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Return:

NO RESPONSE TEXT

[deassignSituation](#)

deassignSituation

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

getActiveSituationIds

getActiveSituationIds

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

Request Argument

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

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Example

Successful request return:

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

getPrcLabels

getPrcLabels

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

Request Arguments

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

Return Parameter

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

HTTP code	HTTP status or error code indicating request success or failure For codes, see below
-----------	---

Example

Curl Command:

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

Successful Return:

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

getSigCorrelationInfo

getSigCorrelationInfo

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

```
[
  {
    "sig_id": 1,
```

```

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

```

getSituationActions

getSituationActions

A GET request that returns the activities for Situations, ordered most recent last

Request Arguments

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request
sitn_ids	Array	Array of Situations that the activities are asked for
start	Integer	Starting from which row should data be included in results
limit	Integer	Limit for how many activities wanted in the output
actions	Array	List of actions_code

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

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

Name	Type	Description
uid	Number	User ID
action_code	Number list	Code for the action in that JSON object
description	String	Description of the action
details	String	Details of the action
type	String	Type of the action
sig_id	Integer	The situation ID
timed_at	Integer	Time stamp of the activity

Example

Curl Command

```


```

```
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'
```

Successful request return:

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

getSituationAlertIds

getSituationAlertIds

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

Request Arguments

Name	Type	Description
auth_token	String	A valid auth_token returned from the authentication request
sitn_id	Number	The Situation ID
for_unique_alerts	Boolean	Indicates the alerts to get from the Situation: true = get only alerts unique to the Situation false = get all alerts in the Situation (default)

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Example

Curl Command

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

Successful request return:

```
{"total_alerts":232,"alert_ids": [6,10,17,19,22,26,27,29,32,43,44,45,47,52,67,68,79,81,83,84,96,102,105,1
08,109,111,113,115,116,125,135,136,138,140,142,143,147,151,152,153,165,1
75,177,178,180,181,188,192,193,207,211,213,217,223,225,232,238,239,240,2
44,255,258,259,269,270,272,274,284,293,303,314,318,335,357,363,369,374,3
75,388,398,414,428,430,434,442,443,448,449,450,479,480,485,486,492,494,5
04,505,510,511,518,521,529,556,558,563,570,580,594,596,599,601,603,628,6
55,656,661,664,674,684,691,705,714,715,719,720,728,732,734,750,776,777,7
81,788,794,808,819,830,835,838,844,857,858,860,861,877,882,885,887,890,8
92,893,900,901,906,912,914,918,926,936,937,959,971,972,984,994,1004,1013
,1016,1019,1020,1023,1033,1043,1045,1068,1076,1082,1083,1085,1099,1119,11
24,1135,1137,1143,1147,1171,1185,1201,1207,1217,1225,1231,1238,1254,1271
,1272,1274,1280,1282,1290,1292,1301,1320,1321,1322,1324,1326,1327,1331,13
32,1333,1362,1379,1402,1414,1423,1433,1443,1454,1468,1472,1473,1481,1491
,1510,1512,1517,1520,1522,1532,1534]}
```

getSituationDescription

getSituationDescription

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Example

Curl Command

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

Successful request return:

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

getSituationDetails

getSituationDetails

A GET request that returns details (such as Description, Status, etc.) of a specified Situation.

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

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

Name	Type	Description
category	String	The Situation category. One of the following: <ul style="list-style-type: none">• Closed• Created• Detected• Priority• Spam• Superseded
created_at	Number	The timestamp (in Unix epoch time) of the Situation creation time
custom_info	JSON Object	A JSON Object containing the custom information
description	String	The Situation description
first_event_time	Number	The timestamp (in Unix epoch time) of the first Event used in creating this Situation
internal_priority	Number	

		The Situation priority, calculated from the priorities of the alerts in the Situation																		
last_event_time	Number	The timestamp (in Unix epoch time) of the last Event used in creating this Situation																		
last_state_change	Number	The timestamp (in Unix epoch time) of the last status change of this Situation																		
moderator_id	Number	The User ID of the user that this the moderator of this Situation																		
sitn_id	Number	The Situation ID																		
status	Number	<p>The Situation's status as an integer:</p> <table border="1"> <tr><td>1</td><td>Opened</td></tr> <tr><td>2</td><td>Unassigned</td></tr> <tr><td>3</td><td>Assigned</td></tr> <tr><td>4</td><td>Acknowledged</td></tr> <tr><td>5</td><td>Unacknowledged</td></tr> <tr><td>6</td><td>Active</td></tr> <tr><td>7</td><td>Dormant</td></tr> <tr><td>8</td><td>Resolved</td></tr> <tr><td>9</td><td>Closed</td></tr> </table>	1	Opened	2	Unassigned	3	Assigned	4	Acknowledged	5	Unacknowledged	6	Active	7	Dormant	8	Resolved	9	Closed
1	Opened																			
2	Unassigned																			
3	Assigned																			
4	Acknowledged																			
5	Unacknowledged																			
6	Active																			
7	Dormant																			
8	Resolved																			
9	Closed																			
story_id	Number	The ID of the History story for this Situation																		
superseded_by	Number/Null	The ID of a Situation that supersedes this Situation, if there is one. Can be null																		
total_alerts	Number	The number of all alerts in this Situation																		
total_unique_alerts	Number	The number of alerts unique to this Situation																		

Example

Curl Commands

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

Successful request return:

```
{
  "category": "Detected", "created_at": 1415814620,
  "custom_info": null, "description": "Sigaliser
situation", "first_event_time": 1415814600,
  "internal_priority": 0,
  "last_event_time": 1415814619,
```

```

    "last_state_change":1415868947,
    "moderator_id":2,
    "sitn_id":3,
    "status":1,
    "story_id":3,
    "superseded_by":null,
    "total_alerts":1403,
    "unique_alerts":1402
}

```

getSituationHosts

getSituationHosts

A GET request that 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.sourcefield in the database) for the sources of Events.

Request Arguments

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request
sitn_id	Number	The Situation ID
for_unique_alerts	Boolean	Indicates the alerts to use to get the host names: true = use only alerts unique to the Situation false = use all alerts in the Situation (default)

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

Name	Type	Description
hosts	List	A list of the host names

Example

Curl Command

```

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

```

Successful request return:

```
{"hosts":["172.27.127.247-888","90.223.60.20-SSS","bm1.wnwx.isp.acme.
com-OOO","vm3.sshyw.isp.acme.com-OOO"]}
```

getSituationIds

getSituationIds

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Example

Curl Command:

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

Successful request return:

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

SQL-like Filters

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

Example curl request to get the first 20 situation_ids with query: **description = 'test1' OR queue = 5**:

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

getSituationProcesses

getSituationProcesses

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

Name	Type	Description
processes	List	A list of the Situation's process names

Example

Curl Command

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

Successful request return, with a primary process name defined:

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

getSituationServices

getSituationServices

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

Name	Type	Description
services	List	A list of the Situation's service names

Example

Curl Command

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

Successful request return, with a primary service name defined:

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

getThreadEntries

getThreadEntries

A GET request that returns thread entries for a specified Situation. Threads are comments or 'story activity' on Situations (More information [here](#)).

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

Request Arguments

Name	Type	Description
auth_token	String	A valid auth_tokenreturned from the authenticaterequest
sitn_id	Number	The Situation ID
thread_name	String	The name of the thread to get entries from
start	Number	The number of the first thread entry to return (default = 0)
limit	Number	The maximum number of thread entries to return (default = 100)

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

The entries list contains the following sub parameters:

Name	Type	Description
user_id	Number	The user ID of the user that created the entry

time	Number	The timestamp (in Unix epoch time) of when the entry was created
entry_text	String	The text of the entry

Example

Curl Command

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

Successful request return:

```
{"entries": [{"entry_text": "My second entry here", "user_id": 5, "time": 1499424718}, {"entry_text": "My entry here", "user_id": 5, "time": 1499424710}], "sitn_id": 358, "thread_name": "Support"}
```

removeSigCorrelationInfo

removeSigCorrelationInfo

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

Request Arguments

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request.
sitn_id	Number	The Situation ID.
serviceName	String	The service name (Optional)
externalId	String	The external ID (Optional).

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

A successful return is an HTTP 200 response.

resolveSituation

resolveSituation

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

setPrcLabels

setPrcLabels

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

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

Request Arguments

Name	Type	Description
sitn_id	Number	The Situation ID
alert_ids	Number list	A list of the alert IDs
causal	List	PRC levels
non_causal		
unlabelled		

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful return:

NO RESPONSE TEXT

[setSituationAcknowledgeState](#)

setSituationAcknowledgeState

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful return:

NO RESPONSE TEXT

[setSituationDescription](#)

setSituationDescription

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

[setSituationProcesses](#)

setSituationProcesses

A POST request that applies a list of [processes](#) to a specified Situation.

Any other processes already associated with the Situation are removed.

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful return:

NO RESPONSE TEXT

[setSituationServices](#)

setSituationServices

A POST request that applies a list of [external services](#) to a specified Situation.

Any other services already associated with the Situation are removed.

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE TEXT

User Management

[applyNewLicense](#)

applyNewLicense

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object with the following:

Example

Curl Command:

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

Successful request return:

```
HTTP/2 200
```

createTeam

createTeam

A POST request that creates a new team.

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object with the following:

Name	Type	Description
team_id	Number	The new team ID

Example

Curl Command:

```
curl -X POST -u graze:graze -k -v "https://localhost/graze/v1  
/createTeam" -H "Content-Type: application/json; charset=UTF-8" -d '{"name" : "my team name 1",  
"alert_filter" : "manager = \"mv_manager\""}'
```

```

and (class = \"my_class_12345\" or external_id = \"my_ext_12345\"), "services" : ["Identity Management","Yellow Pages"], "sig_filter" : "description = \"my_description_12345\" or queue = 50", "landing_page"
: { "type":"situations", "id":"open" }, "active" : true, "description" : "The team description 12345",
"users" : ["user1","user2","user3"]}'

```

Successful request return:

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

[createUser](#)

createUser

A POST request that creates a new user.

Request Arguments

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request
username	String	Mandatory - the new user (unique) login username
password	String	The new user password (only valid for DB realm)
active	Boolean	true if the user active, false if the user inactive, default to true
email	String	The user email address
fullname	String	The user full name
roles	JSON list	That list should contain either the list the role IDs or the role names. E.g "roles": ["Super User"],
primary_group	String or Number	The user primary group name or primary group ID
department	String or number	The user department ID or name
joined	Number	The time the user joined (in Unix time)
timezone	String	The user timezone
contact_num	String	The user phone number
session_expiry	Number	The number of minutes after which the user session will expire. Default to system default
competencies	JSON list	A list with the user competencies. Each competency should have have name or cid and ranking. That is, something like:

```
[
```

		<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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object with the following:

Name	Type	Description
user_id	Number	The new user ID

Example

Curl Command:

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

Successful request return:

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

getTeam

getTeam

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure. For codes, see below.

Example

Curl Commands:

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

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

Successful request return:

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

↳ [getTeamsForService](#)

getTeamsForService

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure. For codes, see below .

Example

Curl Commands:

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

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

Successful request return:

```
[  
  {  
    "room_id": 1, "alert_filter":  
      "", "user_ids": [  
        3  
      ],  
      "sig_filter": "", "name": "Cloud  
DevOps", "landing_page": "",  
      "description": "Example Team", "active":  
        true,  
      "service_ids": [ 1,  
        2,  
        3,  
        4,  
        5,  
        6,  
        7,  
        8,  
        9,  
        10,
```

```

    11
],
"team_id": 1,
"services": [
    "Commerce",
    "Compute",
    "CRM",
    "Database",
    "Mobile",
    "Networking",
    "Remote",
    "Social",
    "Storage",
    "Switch",
    "Web"
],
"users": [
}
]

```

[getTeamSituationIds](#)

getTeamSituationIds

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

Request Argument

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

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Example

Curl Command

```


```

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

Successful request return:

```
{"total_situations":35,"sitn_ids": [20,21,39,55,85,119,145,180,208,233,244,275,305,358,460,461,485,518,574, 575,592,616,666,695,696,740,800,892,919,960,992,993,1027,1047,1092]}
```

[getTeamSituationStats](#)

getTeamSituationStats

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

Request Argument

Name	Type	Description
teams	Array	An array of team IDs - Optional : If no teams are provided, the endpoint will return data for the top 10 teams
from	Number	A timestamp from epoch in seconds - The time from when the data points will be collected
to	Number	A timestamp from epoch in seconds - The time until when the data points will be collected

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

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

Output Example

Successful request return:

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

getuserInfo

getUserInfo

A GET request that returns information about a specified user.

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Example

Curl Command

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

Successful request return:

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

getUserRoles

getUserRoles

Fetches the user's roles from the database.

Request Argument

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

Return parameter

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

Example

Curl Command:

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

Successful request return:

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

getUsers

getUsers

Fetches a list of all users in the database.

Request Argument

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

Return parameter

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

Example

Curl Command:

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

Successful request return:

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

↳ [getUserTeams](#)

getUserTeams

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

Request Argument

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

Return parameter

Type	Description
NativeObject	A Javascript object containing Team ID and Team name

Example

Curl Command:

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

Successful request return:

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

updateTeam

updateTeam

A POST request that modify an existing team.

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

[updateUser](#)

updateUser

A POST request that modify an existing user.

Request Arguments

Name	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request
username	String	Mandatory (optional if user ID used) - username for user to be edited
uid	Long	Mandatory (optional if username used) - user ID for user to be edited
password	String	Optional - The new user password (only valid for DB realm)
active	Boolean	Optional - true if the user active, false if the user inactive, default to true
email	String	Optional - The user email address
fullname	String	Optional - The user full name
roles	JSON list	Optional - That list should contain either the list the role IDs or the role names. E.g "roles":["Super User"],
primary_group	String or Number	Optional - The user primary group name or primary group ID
department	String or number	Optional - The user department ID or name
timezone	String	Optional - The user timezone
contact_num	String	Optional - The user phone number
session_expiry	Number	Optional - The number of minutes after which the user session will expire. Default to system default

competencies	JSON list	Optional - A list with the user competencies. Each competency should have name or cid and ranking. That is, something like:
teams	JSON list of numbers or strings	Optional - List of the user teams. The list should contains either the list of the teams ID or the teams name

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful request return:

NO RESPONSE TEXT

 [authenticate](#)

authenticate

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

Request Arguments

Name	Type	Description
username	String	A valid Cisco Crosswork Situation

	Manager username
--	------------------

password	String	The username's corresponding password
----------	--------	---------------------------------------

Return Parameter

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

Example

Curl Command

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

Return

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

All requests (other than authenticate) require a valid auth_token or basic authentication header. Therefore before any Graze API request is used, a valid authenticate request must be successfully made unless basic authentication headers will be 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 are making regular Graze requests within a one hour timeframe you are considered active and your session will not expire

Security Realms

createSecurityRealm

createSecurityRealm

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

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

Request Arguments

Name	Type	Description
name	String (Required)	Name of the security realm.
type	String (Required)	Security realm type. This must be 'SAML2'.
active	Boolean (Required)	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 (Required)	

JSON object containing the realm configuration. For reference see [Security Configuration Reference](#).

Upload your IdP metadata file using `idpMetadata` or specify the location of the file using `idpMdUrl`. For example:

- "idpMetadataUrl":"http://<location_of_idp_metadata>"
- "idpMetadata":"<raw_ipd_metadata.xml>"

Response

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below .

Example

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

```
curl -X POST \
      -u graze:graze \
      -k -v "https://localhost/graze/v1/createSecurityRealm" \
      -d {"name": "mySamlRealm", "type": "SAML2", "active": "true", "configuration": {
        "idpMetadataUrl": "http://exampleIdP:18080/auth/realms/master/protocol/saml/descriptor",
        "defaultRoles": ["Operator"],
        "defaultTeams": ["Cloud DevOps"],
        "defaultGroup": "End-User",
        "existingUserMappingField": "username",
        "username": "$username", "email": "$email",
        "fullname": "$firstname $lastname",
        "maximumAuthenticationLifetime": 60
      }
    }
```

A successful request returns an HTTP 200 response.

getSecurityRealm

getSecurityRealm

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

Request Arguments

None required.

Response

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below .

Example

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

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

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

```
{  
    "Test Saml Realm": {  
        "configuration": {  
            "defaultGroup": "EndUser", "realmType": "SAML2",  
            "existingUserMappingField": "username",  
            "spMetadataFile": "/usr/share/moogsoft/config  
/keycloak.my_sp_metadata.xml",  
            "defaultRoles": ["Operator"], "defaultTeams": ["Cloud DevOps"],  
            "fullname": "$FirstName $LastName", "email": "$Email",  
            "idpMetadataFile": "/usr/share/moogsoft/config  
/keycloak.my_idp_metadata.xml",  
            "username": "$Email",  
            "maximumAuthenticationLifetime": 60}, "name": "Test  
Saml Realm", "active": true,  
            "type": "SAML2"  
        }  
    },  
    "Google realm": {  
        "configuration": {  
            "realmType": "GOOGLE"}, "name": "Google  
realm", "active": true, "type": "GOOGLE"  
    }  
}
```

updateSecurityRealm

updateSecurityRealm

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

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

Request Arguments

Name	Type	Description
name	String (Required)	Name of the security realm.
type	String	Security realm type. This must be 'SAML2'.
active	Boolean	Determines whether the new realm is

		active or not.
configuration	JSON Object	JSON object containing the realm configuration. You must include all mandatory configuration properties; otherwise the request returns an error. See Security Configuration Reference .

Response

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below .

Example

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

```
curl -X POST \
  -u graze:graze \
  -k -v "https://localhost/graze/v1/updateSecurityRealm" \
  -d {"name": "mySamlRealm"
  -d 'configuration=
  {
    "idpMetadata": "<?xml version='1.0' encoding='UTF-8'?>
<r>n<EntitiesDescriptor Name='urn:keycloak' xmlns='urn:oasis:names:tc:SAML:2.0:metadata'><r>n<EntityDescriptor entityID='http://moogsaml:18080/auth/realm/master'><r>n<IDPSSODescriptor WantAuthnRequestsSigned='true'><r>n<protocolSupportEnumeration='urn:oasis:names:tc:SAML:2.0: protocol'><r>n<KeyDescriptor use='signing'><r>n<dsig: KeyInfo><r>n<dsig: KeyName>l8ddhI8SroeNnlq0TkTxIj2VI-0bvr2QfG_o32jWeKI</dsig: KeyName><r>n<dsig: X509Data><r>n<dsig: X509Certificate>MIICmzCCAYMCBgFk8A9vMjANBqkhkiG9w0BAQsFADARMQ8wDQYDVQQDDAZtYXN0ZXIwHhcNMTgwNzMxMTEExNjQwWhcNMjgwNzMxMTExODIwWjARMQ8wDQYDVQQDDAZtYXN0ZXIwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCOliZ3dBu696sIYduAb1BMuvR1bMdTKVBMICWaEEcS8Rzw8gWthPQpw2e202LjOeu4VkJVmEEAUa2IrLS4QpYgyhOuzapcIGF4kB0ARebalWa7C9od%2BeTqWgvXPrDOKzp7g%2B%2Ba5yvtKxE3ieUORPpACvLWcbkMwyb%2Be5V8%2Bz8n4263Uol8srSaxLsm/oTozJNwbG%2BbzV8JQHU3xFV5nFbyNySvc%2B%2B7tDFZuJC5BMu6bwi/rPqp5OMcuB1W%2BxCcX7IYPphnBjRWNyQJD3gRCKjrujISkTEcqZEjR79isbofQaPDi5TSjglPD5rr0OWMVqv91a1/pVN2y0y%2B%2BRIT8HAgMBAAEwDQYJKoZIhvcNAQELBQADggEBAgRhWYKESVsTRAUVYzHYptd3/eX47%2BTVXhjPO0ORLUJbHtfhgohtyejd6ohazkcSgMy6%2BwaeVojqq4Q%2B%2tzCOW2EAqO9QOQdaBWOPxDXhJ9TGQJE2E28SS2Gg6paAMfRmtA7c6xXii%2BYfLo3PG1SSc/sGe4KIPKflkqqDEqEeaY1oIPZU2bLnpMSIui2nK1crE2%2Bt9apLWAGosah6scMGZ9vTrtOVrNuhB2LuU3cvRQWrUBaQuXQsBV7Q6a8lkrrZ6rjAIbO4vcEL4yjQpnA%2BhetuhBlGPQj6ntuhdnmoKmWYY97wk8eXwblhQxg8GUyfqabfOAKwiGAklxgkexm20M=<\
```

```
/dsig:X509Certificate>|r\n</dsig:X509Data>|r\n</dsig:KeyInfo>|r\n<
/KeyDescriptor>|r\n|r\n<SingleLogoutService|r\nBinding=|"urn:oasis:
names:tc:SAML:2.0:bindings:HTTP-POST\"|r\nLocation=|"http://moogsaml:
18080/auth/realms/master/protocol/saml" \
/>|r\n<SingleLogoutService|r\nBinding=|"urn:oasis:names:tc:SAML:2.0: bindings:HTTP-
Redirect"|r\nLocation=|"http://moogsaml:18080/auth\
```

```

/realmss/master/protocol/saml\" />|r\n<NameIDFormat>urn:oasis:names: tc:SAML:2.0:nameid-
format:persistent<|
/>|r\n<NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:transient</NameIDFormat>|r\n<NameIDFormat>urn:oasis:names:tc: SAML:1.1:nameid-
format:unspecified</NameIDFormat>|r\n<NameIDFormat>urn: oasis:names:tc:SAML:1.1:nameid-
format:emailAddress<|
/>|r\n<SingleSignOnService Binding=|"urn:oasis:names:tc:
SAML:2.0:bindings:HTTP-POST"\r\nLocation=|"http://moogsaml:18080\
/auth/realms/master/protocol/saml"\ |
/>|r\n<SingleSignOnService\r\nBinding=|"urn:oasis:names:tc:SAML:2.0: bindings:HTTP-
Redirect"\r\nLocation=|"http://moogsaml:18080/auth\
/realms/master/protocol/saml"\ |
/>|r\n<SingleSignOnService\r\nBinding=|"urn:oasis:names:tc:SAML:2.0:
bindings:SOAP"\r\nLocation=|"http://moogsaml:18080/auth/realms\
/master/protocol/saml"\ |>|r\n</IDPSSODescriptor>r\n<
/EntityDescriptor>r\n</EntitiesDescriptor>,
    "defaultRoles": ["Operator"], "defaultTeams": ["Cloud
    DevOps"], "defaultGroup": "End-User",
    "existingUserMappingField": "username",
    "username": "$username",
    "email": "$email", "fullname": "$firstname
    $lastname", "maximumAuthenticationLifetime": 60
  }'

```

Curl command to deactivate an active SAML realm:

```

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

```

Successful return:

A successful request returns an HTTP 200 response.

Dashboards and Reporting

[getTeamSituationStats](#)

getTeamSituationStats

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

Request Argument

Name	Type	Description
teams	Array	An array of team IDs - Optional : If no teams are provided, the endpoint will return empty data
from	Number	A timestamp from epoch in seconds - The time from when the data points will be collected

to	Number	A timestamp from epoch in seconds - The time until when the data points will be collected
----	--------	---

aggregation	String	Can be "none" (No aggregation of results) or "sum" (Aggregation of all teams provided)
-------------	--------	--

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

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

Output Example

Successful request return:

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

getSystemSituationStats

getSystemSituationStats

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

Request Argument

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

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

Output Example

Successful request return:

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

```

        ],
        "target": "Open Situations"
    }]
}

```

getMTTRStats

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

Request Argument

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

```

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

```

Output Example

Successful request return:

```

[{
    "datapoints": [
        [2.0, 1513657700000],
        [9.0, 1513661300000],
        [1.0, 1513664900000],
        [4.0, 1513668500000],
        [3.0, 1513672100000],
        ...
    ]
}
]

```

```

        [2.0, 1513675700000],
        [2.0, 1513679300000],
        [9.0, 1513682900000],
        [1.0, 1513686500000]
    ],
    "target": "mtt_resolve"
}

```

[getMTTDStats](#)

getMTTDStats

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

Request Argument

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

```

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

```

Output Example

Successful request return:

```

[{
    "datapoints": [
        [2.0, 1513657700000]
    ]
}
]

```

```

        [9.0, 1513661300000],
        [1.0, 1513664900000],
        [4.0, 1513668500000],
        [3.0, 1513672100000],
        [2.0, 1513675700000],
        [2.0, 1513679300000],
        [9.0, 1513682900000],
        [1.0, 1513686500000]
    ],
    "target": "mtt_detect"
}

```

getMTTASStats

getMTTASStats

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

Request Argument

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

```

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

```

Output Example

Successful request return:

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

[getReoccurringSituationStats](#)

getReoccurringSituationStats

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

Request Argument

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

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

Output Example

Successful request return:

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

getReassignedSituationStats

getReassignedSituationStats

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

Request Argument

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

```
[
```

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

Output Example

Successful request return:

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

getServiceSituationStats

getServiceSituationStats

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

Request Argument

Name	Type	Description
services	Array	An array of services IDs - Optional : If no services are provided, the endpoint will return empty data
from	Number	A timestamp from epoch in seconds - The time from when the data points will be collected
to	Number	A timestamp from epoch in seconds - The time until when the data points will be collected
aggregation	String	Can be "none" (No aggregation of results) or "sum" (Aggregation of all services provided)

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

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

Output Example

Successful request return:

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

getSituations

getSituations

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

Request Argument

Name	Type	Description
status	Array	An array of status IDs. Optional, if not given, returns default set of statuses: Opened, Unassigned, Assigned, Acknowledged, Unacknowledged, Resolved.
from	Number	A timestamp from epoch in seconds - The time from when the data points will be collected

to	Number	A timestamp from epoch in seconds - The time until when the data points will be collected
aggregation	String	Can be "none" (No aggregation of results) or "sum" (Aggregation of all statuses provided)

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

```
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'
```

Output Example

Successful request return:

```
[{
    "target": "situation",
    "datapoints": [
        [92.0, 1516008478000],
        [666.0, 1516030078000]
    ],
    "status": "OK"
}, {
    "target": "status",
    "datapoints": [
        [1.0, 1515947278000],
        [35.0, 1515958078000],
        [63.0, 1515976078000],
        [241.0, 1515994078000],
        [4.0, 1516015678000]
    ],
    "status": "OK"
}]
```

↳ [getSeveritySituationStats](#)

getSeveritySituationStats

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

Request Argument

Name	Type	Description
severity	Array	An array of severity IDs. Optional, if not given, returns default set of severities: Clear, Indeterminate, Warning, Minor, Major, Critical.
from	Number	A timestamp from epoch in seconds - The time from when the data points will be collected
to	Number	A timestamp from epoch in seconds - The time until when the data points will be collected
aggregation	String	Can be "none" (No aggregation of results) or "sum" (Aggregation of all severities provided)

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

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

Output Example

Successful request return:

```
[{
  "datapoints": [
    192.0, 1516008478000]
```

```

        [666.0, 1516030078000]
    ],
    "target": "Major"
}, {
    "datapoints": [
        [1.0, 1515947278000],
        [35.0, 1515958078000],
        [63.0, 1515976078000],
        [241.0, 1515994078000],
        [4.0, 1516015678000]
    ],
}
]
```

Workflow

[addProcess](#)

addProcess

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

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

```

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

Successful return:

NO RESPONSE TEXT

[addService](#)

addService

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

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

Request Arguments

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

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Return:

NO RESPONSE
TEXT

createMaintenanceWindow

createMaintenanceWindow

A POST request that creates a maintenance window period that filters alerts.

Request Argument

Parameter	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request
name	String	The name of the window
description	String	Description of the window
filter	String	JSON or SQL like filter for which alerts to match. The filter must be in the correct Moog JSON format. i.e. the same format as used in alert and Situation filters in the DB
start_date_time	Epoch Seconds (Number)	Time to start the window from. The start_date_time property must be in epoch time and may be up to 5 years in the future
duration	Seconds (Number)	Duration of the window in seconds. The minimum duration is 1 second and the maximum is 5 years
forward_alerts	boolean	Set this to True if the alert in the maintenance mode should be forwarded to the next Mooleet in the chain

recurring_period	Number	How many days/weeks/months to wait
------------------	--------	------------------------------------

		before this recurs. The recurring_period property must be 1 - no other value will be accepted
recurring_period_units	Number	Decides what the recurring period counts in 0 = minutes, 1 = hours, 2 = days, 3 = weeks, 4 = months. The recurring_period_units property has allowed values of 2 (daily), 3 (weekly) or 4 (monthly) - no other values will be accepted

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

SQL-like Filters

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

The 'filter' parameter now accepts a string (SQL-format filter) instead of an object (old style JSON-format filter).

Curl Command:

Example curl request to create a window with filter: **source = 'server1' AND external_id IN ('value1', 'value2', '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}'
```

Successful requests return a JSON object with the following:

Name	Type	Description
window_id	Number	The new Situation ID

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

deleteMaintenanceWindows

deleteMaintenanceWindows

A POST requests that deletes a maintenance window.

Parameter	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request
id	Number	The ID of the window to delete

Return Parameter

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful Command:

NO RESPONSE TEXT

[getIntegrationConfig](#)

getIntegrationConfig

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

The exported JSON file can be saved as a duplicate configuration of the original integration.

For example, you can modify and save the returned object as webhook_lam_custom.conf. Run it with this command:

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful requests return a JSON object:

```
{
```

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

```

```
    "$agent_location"  
},  
{  
    "name": "type",
```

```

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

```

```
"INDETERMINATE": 1,  
"1": 2  
}  
}  
}
```

```
}
```

getMaintenanceWindows

getMaintenanceWindows

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

Request Arguments

Parameter	Type	Description
auth_token	String	A valid auth_token returned from the authenticate request
start	Number	The start point for where to fetch windows from (ie, 0 to start at the first, 10 to start at the 11th)
limit	Number	The number of windows to fetch

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful request return:

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

findMaintenanceWindows

findMaintenanceWindows

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

Request Arguments

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Example

Curl Command:

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

Successful request return:

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

getSeverities

getSeverities

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

No Requested Arguments

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

Name	Type	Description
severity_id	Number	The ID of the severity
name	String	The severity name

Input Example

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

Output Example

Successful request return:

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

getProcesses

getProcesses

A GET request that returns the list of processes.

Request Arguments

Name	Type	Description
auth_token	String (Mandatory)	A valid auth_token returned from the authentication request
limit	Integer (Optional)	The maximum number of results that are returned. Default is 1000.

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

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

Input Example

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

Output Example

Successful request return:

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

getServices

getServices

A GET request that returns the list of services.

Request Arguments

Name	Type	Description
auth_token	String (Mandatory)	A valid auth_token returned from the authenticate request

limit	Integer (Optional)	The maximum number of results that are returned. Default is 1000.
-------	-----------------------	---

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

Name	Type	Description
service_id	Number	The ID of the service
service_name	String	The service name

Input Example

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

Output Example

Successful request return:

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

getStats

getStats

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

No Requested Arguments

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Input Example

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

Output Example

Successful request return:

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

"from": {
seconds", "description": "A timestamp from epoch in

```

        "type": "Long",
        "required": true
    },
    "services": {
        "mapping": {
            "display_value": "name", "endpoint": "getServices", "value": "service_id"
        },
        "type": "mapped",
        "required": false
    },
    "to": {
        "description": "A timestamp from epoch in "type": "Long",
        "required": true
    seconds",
}

},
{
    "endpoint": "getSystemSituationStats",
    "description": "returns the number of active situations in the
system over time",
    "display_name": "Open Situations", "parameters": {
    {
        "from": {
            "description": "A timestamp from epoch in
seconds",
            "type": "Long",
            "required": true
        },
        "to": {
            "description": "A timestamp from epoch in "type": "Long",
            "required": true
    seconds",
}

},
{
    "endpoint": "getStatusSituationStats",
    "description": "returns the number of active situations with
specified status over time",
    "display_name": "Situations by Status", "parameters": {
        "teams": {
            "mapping": {
                "display_value": "name",

```

```
        "endpoint": "getTeams", "value":  
        "team_id"  
    },  
    "type": "mapped", "required": false  
},
```

```

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

```

```
" required": true  
},  
"to": {
```

```

        "description": "A timestamp from epoch in
seconds",
        "type": "Long",
        "required": true
    }
}

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

```

```
"from": {  
    "description": "A timestamp from epoch in \"type\": \"Long\",  
    seconds",
```

```

        "required": true
    },
    "to": {
        "description": "A timestamp from epoch in \"type\": \"Long\"",
        "required": true
    seconds",
}

    }
}

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

}, {
    "endpoint": "getReassignedSituationStats",
    "description": "returns the number of situations that have
been reassigned over time",
    "display_name": "Reassigned situations", "parameters": {
        "teams": {
            "mapping": {
                "display_value": "name", "endpoint":
                "getTeams", "value": "team_id"
            },
            "type": "mapped", "required": false
        }
    }
}

```

```
"type ": "mapped ",  
"required ": false  
},  
"from ": {
```

```

        "description": "A timestamp from epoch in
seconds",
        "type": "Long",
        "required": true
    },
    "to": {
        "description": "A timestamp from epoch in
seconds",
        "type": "Long",
        "required": true
    }
}
]

```

getStatuses

getStatuses

A GET request that returns the list of active teams.

No Requested Arguments

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

Name	Type	Description
status_id	Number	The ID of the status
name	String	The status name

Input Example

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

Output Example

Successful request return:

```
[
    {
        "status_id": 1,
        "name": "Onened"
    },
    {
        "status_id": 2,
        "name": "Unassinged"
    }
]
```

```

        "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"
    }
]
```

[getSystemSummary](#)

getSystemSummary

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

Request Argument

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

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

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

Name	Type	Description
open_sitns	Number	The number of open Situations in Cisco Crosswork Situation Manager
open_sitns_down	Number	The number of open Situations that are trending down
open_sitns_up	Number	The number of open Situations that are trending up
avg_events_per_sitn	Number	The average number of events per situation
avg_alerts_per_sitn	Number	The average number of alerts per situation

service_count	Number	The number of services currently in Cisco Crosswork Situation Manager
open_sigs_unassigned	Number	The number of situations unassigned
total_events	Number	The total number of Events currently in Cisco Crosswork Situation Manager

Example

Curl Command

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

Successful request return:

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

getSystemStatus

getSystemStatus

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

Request Argument

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

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

Return Parameters

Type	Description
HTTP code	HTTP status or error code indicating request success or failure For codes, see below

Successful requests return a JSON object containing the following:

Name	Type	Description
component	String	Represents the name of a component within the process. May not be present, depending on the process
instance	String	The instance name
last_heartbeat	Number	The timestamp (in Unix epoch time) of the last process heartbeat. 0 is a special value indicating that a heartbeat has never been received
missed_heartbeats	Number	The number of missed process heartbeats. -1 is a special value indicating that a heartbeat has never been received

process_name	String	The process name
processes	List	A list of the processes, with status information
reserved	Boolean	Indicates whether the process is reserved: true = a reserved process false = process not reserved A reserved process is a process that is usually required for Cisco Crosswork Situation Manager to be working properly
running	Boolean	Indicates whether the process is running or not: true = running false = not running
service_name	String	The service name
display_name	String	The name of the service in the configuration
type	String	The type of the service (e.g lam, servlets, moog_farmd)
passive	Boolean	Indicates whether the service is passive in a HA environment: true = passive false = active
stoppable	Boolean	Indicates whether the service can be stopped or not: true : stoppable false : not stoppable
ha_conf	JSON Object	A Json blob containing the HA configuration

Example

Curl Command

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

Successful request return:

```
{
  "processes": [
    {
      "running": true,
      "sub_components": {
        "moogpoller": {
          "run_on_startup": true, "instance": "",
          "service_name": "apache-tomcat",
          "display_name": "servlets", "type": "servlets",
          "last_heartbeat": 1491385834300,
          "passive": false
        }
      }
    }
  ]
}
```

```
        "running": true, "component":  
        "moogpoller", "reserved": true,  
        "stoppable": true, "missed_heartbeats":  
        0, "ha_conf": {  
            "cluster": "MOO",  
            "instance": "", "default_leader": true,  
            "start_as_passive": false,  
            "only_leader_active": false, "group":  
            "servlets"  
        }  
    },  
    "moogsvr": {  
        "run_on_startup": true, "instance":  
        "",  
        "service_name": "apache-tomcat",  
        "display_name": "servlets", "type": "servlets",  
        "last_heartbeat": 1491385825246, "passive":  
        false,  
        "running": true, "component":  
        "moogsvr", "reserved": true,  
        "stoppable": true,  
        "missed_heartbeats": 0,  
        "ha_conf": {  
            "cluster": "MOO",  
            "instance": "", "default_leader": true,  
            "start_as_passive": false,  
            "only_leader_active": false, "group":  
            "servlets"  
        }  
    },  
},  
"instance": "",  
"reserved": true,  
"service_name": "apache-tomcat", "stoppable":  
true, "missed_heartbeats": 0, "display_name":  
"servlets", "type": "servlets", "last_heartbeat":  
1491385834300, "ha_conf": {  
    "cluster": "MOO",  
    "instance": "", "default_leader": true,  
    "start_as_passive": false,  
    "only_leader_active": false, "group":  
    "servlets"  
},
```

```

    "passive": false
  },
  "running": false,
  "instance": "",
  "last_missed_heartbeat": 1491385820601, "reserved":
  false,
  "stoppable": false, "missed_heartbeats":
  10, "display_name": "test_lam", "type":
  "lam",
  "last_heartbeat": 1491382820601,
  "additional_health_info": {
    "thread_pool_queue_size": 0,
    "published_events": {
      "last_5_minutes": 130,
      "last_10_minutes": 130,
      "last_minute": 130
    }
  },
  "ha_conf": {
    "cluster": "MOO",
    "instance": "", "default_leader": true,
    "start_as_passive": false,
    "only_leader_active": true, "group":
    "test_lam"
  },
  "passive": false
},
{
  "sub_components": {
    "SituationMgr": {
      "run_on_startup": true, "instance":
      "",
      "last_missed_heartbeat": 1491385821669,
      "service_name": "moogfarmd", "display_name":
      "moog_farmd",
      "type": "moog_farmd", "last_heartbeat":
      1491382821669, "passive": false,
      "running": false, "component":
      "SituationMgr", "reserved": true,
      "stoppable": true, "missed_heartbeats":
      10, "ha_conf": {
        "cluster": "MOO",
        "instance": "", "default_leader": true,
        "start_as_passive": false,
        "only_leader_active": true, "group":
        "moog_farmd"
      }
    }
  }
}

```

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

```

        "additional_health_info": { "event_processing_metric":  

          0.65  

        },  

        "passive": false, "running": false,  

        "reserved": true, "stoppable": true,  

        "missed_heartbeats": 10, "ha_conf":  

        {  

          "cluster": "MOO",  

          "instance": "", "default_leader": true,  

          "start_as_passive": false,  

          "only_leader_active": true, "group":  

          "moog_farmd"  

        }  

      }, {  

        "running": false, "instance": "",  

        "reserved": false,  

        "service_name": "restclientlamd", "stoppable":  

        true,  

        "display_name": "rest_client_lam", "type": "lam",  

        "ha_conf": {  

          "cluster": "MOO",  

          "instance": "",  

          "group": "rest_client_lam"  

        }  

      }]  

    }
  
```

POST parameters

form-urlencoded

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

Curl Command:

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

application/json

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

Curl Command:

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

HTTP status and error codes

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

HTTP Code	Meaning
200	Successful request
400	Incorrectly formatted request
401	A request with an invalid or expired auth_code
403	Forbidden request
500	Failed request, for example due to an invalid sitn_id

Situation Action Codes

The getSituationActions Graze endpoint 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	Stats API
1	Situation Created	You can use the Stats API endpoints to report on Cisco Crosswork Situation Manager data.
2	Assigned Moderator	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).
3	Situation Resolved	
4	Situation Revived	Endpoints
5	Situation Closed	getAlertsInNewSituations
6	Assigned Queue	getAlertsInNewSituationsStats
7	Created By Merge	A GET request that returns the number of alerts that belong to new Situations during a given time range.
8	Used In Merge	
9	Created By Split	Request Arguments
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

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Alerts in new situations"
datapoints	Number array	An array of data points. Each data point is an array in the format [datapoint, timestamp]: 1. Data point - Number of alerts. 2. Timestamp - Time of the calculation (UNIX epoch timestamp in milliseconds).

The time range you provide determines the number of data points per time unit the request returns. For example:

- Less than 1 week – Returns the number of alerts for the past hour
- 1 week to 1 month – Returns the number of alerts for the past day
- 1 month to 1 year – Returns the number of alerts for the past week
- More than 1 year – Returns the number of alerts for the past month.

Request Example

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

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

Response Example

A successful response indicating there were 56 alerts at 13:23pm on Wednesday 19th September 2018:

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

getCommentCountPerTeamStats

getCommentCountPerTeamStats

A GET request that returns the total number of comments each hour for a specific team or teams in a given time range.

Request Arguments

Name	Type	Description
teams	Array	An array of team IDs. This is required. If no teams are provided, the endpoint does not return any data.
from	Number	Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.

aggregation	String	"sum" - for the sum of the number of the comments for all teams in the array. "none" - for no aggregation of results.
-------------	--------	--

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	Name of the team.
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Number of comments. 2. Timestamp – calculation time (UNIX epoch timestamp in milliseconds) <p>The time range you provide determines the number of data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of comments for the past hour • 1 week to 1 month – Returns the number of comments for the past day • 1 month to 1 year – Returns the number of comments for the past week • More than 1 year – Returns the number of comments for the past month.

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],
      ...
    ]},
    ...
  ]
}
```

[getMTTAPerTeamStats](#)

getMTTAPerTeamStats

A GET request that returns the mean time to acknowledge (MTTA) a Situation per team for a given time range.

Request Arguments

Name	Type	Description
teams	Array	An array containing a single team ID. You can only provide one team ID for this endpoint. This is mandatory. If no teams are provided, the endpoint does not return any data.
from	Number	Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	"sum" - for an aggregation of all MTTA times. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Mean Time to Acknowledge (MTTA)"
datapoints	Number array	An array of data points. Each data point is an array in the format [data point, timestamp]: 1. Datapoint – MTTA (seconds) for that bucket 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds)

The time range you provide determines number of the data points per time unit the request returns. For example:

- Less than 1 week – Returns the MTTA for the past hour
- 1 week to 1 month – Returns the MTTA for the past day
- 1 month to 1 year – Returns the MTTA for the past week
- More than 1 year – Returns the MTTA for the past month.

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)"
}]
```

getMTTASstats

getMTTASstats

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

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

Request Arguments

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Mean Time to Acknowledge (MTTA)"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ul style="list-style-type: none">1. Data point – MTTA (seconds) for that bucket.2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds) <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none">• Less than 1 week – Returns the MTTA for the past hour• 1 week to 1 month – Returns the MTTA for the past day• 1 month to 1 year – Returns the MTTA for the past week• More than 1 year – Returns the MTTA for the past month.

Request Example

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

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1/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 Acknowledges (MTTA)"
}

```

getMTTDStats

getMTTDStats

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

Request Arguments

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Mean Time to Detect (MTTD)"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – MTTD (seconds) for that bucket 2. Timestamp – calculation time (UNIX epoch timestamp in milliseconds) <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the MTTD for the past hour • 1 week to 1 month – Returns the MTTD for the past day • 1 month to 1 year – Returns the MTTD for the past week • More than 1 year – Returns the MTTD for the past month.

Request Example

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

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

Response Example

Successful request returns an MTTD of 4.53 minutes:

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

getMTTRPerTeamStats

A GET request that returns the mean time to resolve (MTTR) a Situation per team for a given time range.

Request Arguments

Name	Type	Description
teams	Array	An array containing a single team ID. You can only provide one team ID for this endpoint. This is mandatory. If no teams are provided, the endpoint does not return any data.
from	Number	Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
aggregation	String	"sum" - for an aggregation of all MTTR times. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Mean Time to Resolve (MTTR)"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – MTTR (seconds) for that bucket 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds) <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the MTTR for the past hour • 1 week to 1 month – Returns the MTTR for the past day • 1 month to 1 year – Returns the MTTR for the past week • More than 1 year – Returns the MTTR for the past month.

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)"
}]
```

↙ [getMTTRStats](#)

getMTTRStats

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

Request Arguments

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Mean Time to Resolve (MTTR)"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [datapoint, timestamp]:</p> <ul style="list-style-type: none"> 1. Data point – MTTR (seconds) for that bucket 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds) <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the MTTR for the past hour • 1 week to 1 month – Returns the MTTR for the past day • 1 month to 1 year – Returns the MTTR for the past week • More than 1 year – Returns the MTTR for the past month.

Request Example

A cURL request to retrieve the MTTR for Cisco Crosswork Situation Manager from 11.30am on Sunday, September 24th 2017 until 11.30am on Sunday, 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)"
}]
```

getNewAlertsStats

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

Request Arguments

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"New Alerts"
datapoints	Number array	An array of data points. Each data point is an array in the format [data point, timestamp]: 1. Data point – Number of alerts. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). The time range you provide determines number of the data points per time unit the request returns. For example: • Less than 1 week – Returns the number of alerts for the past hour

- 1 week to 1 month – Returns the number of alerts for the past day
- 1 month to 1 year – Returns the number of alerts for the past week
- More than 1 year – Returns the number of alerts for the past month.

Request Example

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

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

Response Example

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

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

[getNewAlertsPerSituationsStats](#)

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

Request Arguments

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"New Alerts per Situation"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Percentage noise reduction (alert to Situation reduction). 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the percentage noise reduction for the past hour • 1 week to 1 month – Returns the percentage noise reduction for the past day • 1 month to 1 year – Returns the percentage noise reduction for the past week • More than 1 year – Returns the percentage noise reduction for the past month.

Request Example

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

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

Response Example

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

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

getNewEventsPerAlertsStats

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

Request Arguments

Name	Type	Description
from	Number	Start of the reporting time range. This is a

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

Response

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

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> <ul style="list-style-type: none"> 1. Data point – Percentage noise reduction (event to alert reduction). 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the percentage noise reduction for the past hour • 1 week to 1 month – Returns the percentage noise reduction for the past day • 1 month to 1 year – Returns the percentage noise reduction for the past week • More than 1 year – Returns the percentage noise reduction for the past month.

Request Example

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

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

Response Example

A successful response indicating a 58% noise reduction:

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

[getNewEventsPerSituationsStats](#)

getNewEventsPerSituationsStats

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

Request Arguments

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

Response

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

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> <ol style="list-style-type: none"> 1. Data point – Percentage noise reduction (event to Situation reduction). 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the percentage noise reduction for the past hour • 1 week to 1 month – Returns the percentage noise reduction for the past day

- 1 month to 1 year – Returns the percentage noise reduction for the past week
- More than 1 year – Returns the percentage noise reduction for the past month.

Request Example

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

getNewSituationsStats

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

Request Arguments

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

Response

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

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> <ol style="list-style-type: none"> 1. Data point – Number of new Situations. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past week • More than 1 year – Returns the number of Situations for the past month.

Request Example

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

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

Response Example

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

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

```
        ],
      "target":"New situations"
    }
```

[getReassignedSituationStats](#)

getReassignedSituationStats

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

Request Arguments

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Reassigned Situation"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Number of reassigned Situations. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of reassigned Situations for the past hour • 1 week to 1 month – Returns the number of reassigned Situations for the past day • 1 month to 1 year – Returns the number of reassigned Situations for the past week • More than 1 year – Returns the number of reassigned Situations for the past month.

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"  
  }]  
]
```

↳ [getReassignedSituationsPerTeamStats](#)

getReassignedSituationsPerTeamStats

A GET request that returns the number of reassigned Situations associated with a team or multiple teams over a given time range. A reassigned Situation is a Situation that a user has assigned to another user at least twice.

Request Arguments

Name	Type	Description
teams	Array	An array of team IDs. This is required. If no teams are provided, the endpoint does not return any data.
from	Number	Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
aggregation	String	"sum" - for an aggregation of all Situations. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the team: "<team_name>"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none">1. Data point – Number of reassigned Situations.2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none">• Less than 1 week – Returns the number of reassigned Situations for the past hour• 1 week to 1 month – Returns the number of reassigned Situations for the past day• 1 month to 1 year – Returns the number of reassigned Situations for the past week• More than 1 year – Returns the number of reassigned Situations for the past month.

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]],  
    },  
    {  
      "datapoints": [  
        [5.006, 1533103200000],  
        [5.0, 1533708000000],  
        [5.01, 1534312800000]  
      ]  
    }  
]
```

```

        [5.0714,1534917600000],
        [4.8417,1535522400000]],
      "target":"Application Performance Monitoring"
    []
  }

```

[getReoccurringSituationStats](#)

getReoccurringSituationStats

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

Request Argument

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Reoccurring situations"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Number of reoccurring Situations. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of reoccurring Situations for the past hour • 1 week to 1 month – Returns the number of reoccurring Situations for the past day • 1 month to 1 year – Returns the number of reoccurring Situations for the past week • More than 1 year – Returns the number of reoccurring Situations for the past month.

Request Example

A cURL request to retrieve the number of reoccuring 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 reoccuring Situations during the year:

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

getReoccurringSituationPerTeamStats

A GET request that returns the number of reoccuring Situations associated with a team for a given time range.

Request Argument

Name	Type	Description
teams	Array	An array of team IDs. This is required. If no teams are provided, the endpoint does not return any data.
from	Number	Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
aggregation	String	"sum" - for an aggregation of all Situations. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"Reoccurring situations"

datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Number of reoccurring Situations. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of reoccurring Situations for the past hour • 1 week to 1 month – Returns the number of reoccurring Situations for the past day • 1 month to 1 year – Returns the number of reoccurring Situations for the past week • More than 1 year – Returns the number of reoccurring Situations for the past month.
------------	--------------	--

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"}]
```

↳ [getServiceSituationStats](#)

getServiceSituationStats

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

Request Argument

Name	Type	Description
services	Array	An array of services IDs. This is optional. If no services are provided, the endpoint does not return any data.
from	Number	Start of the time range. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range. This is a Unix epoch timestamp in seconds.

		If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	"sum" - for an aggregation of all teams provided. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	Service name(s).
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Number of Situations impacting services. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past week • More than 1 year – Returns the number of Situations for the past month.

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]
    ],
}]

```

[getServiceSituationPerTeamStats](#)

getServiceSituationPerTeamStats

A GET request that returns the number of Situations impacting each service for a team.

Request Argument

Name	Type	Description
teams	Array	A single team ID in an array. You can only provide one team ID for this endpoint. This is required. If no teams are provided, the endpoint does not return any data.
from	Number	Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
aggregation	String	"sum" - for an aggregation of all services. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the team.
datapoints	Number array	An array of data points. Each data point is an array in the format [data point, timestamp]: 1. Data point – Number of Situations impacting services. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds).

The time range you provide determines number of the data points per time unit the request returns. For example:

- Less than 1 week – Returns the number of Situations for the past hour
- 1 week to 1 month – Returns the number of Situations for the past day
- 1 month to 1 year – Returns the number of Situations for the past week
- More than 1 year – Returns the number of Situations for the past month.

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"
}
```

getSeveritySituationStats

getSeveritySituationStats

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

Request Argument

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the status.
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Number of Situations per severity. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past week • More than 1 year – Returns the number of Situations for the past month.

Input Example

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

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

Output Example

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

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

[getSeveritySituationPerTeamStats](#)

getSeveritySituationPerTeamStats

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

Request Argument

Name	Type	Description
teams	Array	A single team ID in an array. You can only provide one team ID for this endpoint. This is required. If no teams are provided, the endpoint does not return any data.

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the status.
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Number of Situations per severity. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past week • More than 1 year – Returns the number of Situations for the past month.

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]],  
    ]  
}
```

getStats

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

Request Arguments

None required.

Response

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

Request Example

A cURL request to return all available Stats API endpoints:

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

Response Example

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

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



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

```

        "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
    situationsover
    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
        },
        "sort":[]

    }
}
```

```
"to":{  
    "description":"A timestamp from epoch in seconds", "type":"Long",
```

```

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

```



```

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

```

"endpoint":"getReassignedSituationsPerTeamStats", "description":"returns the number of reassigned situations of a team

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

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

"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": "Connection #0 to host freida7 left intact lue":"service_id"
                },
        "type":"mapped",
        "required":"true"
    },
    "to":{
        "description":"A timestamp from epoch in seconds", "type":"Long",
        "required":true
    }
},
{
    "endpoint":"getMTTAPerTeamStats",
    "description":"returns the mean time to acknowledge a situation of a team over time",
    "display_name":"Mean Time To Acknowledge by Team", "parameters":{
        "teams":{
            "mapping":{ "display_value":"name",
                        "endpoint":"getTeams",
                        "value":"team_id"
                    },
            "type":"mapped",
            "required":false
        },
        "from":{
            "description":"A timestamp from epoch in seconds", "type":"Long",
            "required":true
        }
    }
}

```

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

```

"display_value": "Sum",

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



```

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

```

⌄ [getStatusSituationPerTeamStats](#)

getStatusSituationPerTeamStats

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

Request Argument

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

teams	Array	A single team ID in an array. You can only provide one team ID for this endpoint. This is required. If no teams are provided, the endpoint does not return any data.
status	Array	An array of status IDs. Valid status IDs are 1 (opened), 3 (assigned), 4 (acknowledged), 5 (unacknowledged) and 8 (resolved). Other integers do not return any data. If left empty, the request returns the number of Situations for all statuses.
from	Number	Start of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range you want to collect data from. This is a Unix epoch timestamp in seconds.
aggregation	String	"sum" - for an aggregation of all teams provided. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the team.
datapoints	Number array	An array of data points. Each data point is an array in the format [data point, timestamp]: <ol style="list-style-type: none"> 1. Data point – Number of Situations for each status. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). The time range you provide determines number of the data points per time unit the request returns. For example: <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past week • More than 1 year – Returns the number of Situations for the past month.

Request Example

A cURL request with to return all Situations by status for the Cloud DevOps team from 8.30am until 2.30pm on Saturday, September 1st 2018:

```
curl -G -u graze:graze -k -v "https://localhost/graze/v1  
/getStatusSituationPerTeamStats" --data-urlencode 'from=1535790600' -- data-urlencode  
'to=1535812200' --data-urlencode 'teams=[1]' --data- urlencode 'status=[]' --data-urlencode  
'aggregation=none'
```

Response Example

A successful response returns the number of Situations by status each hour for the six hour range:

```
[  
  {"datapoints": [  
    [19.0, 1535790600000],  
    [20.0, 1535794200000],  
    [17.0, 1535797800000],  
    [18.0, 1535801400000],  
    [17.0, 1535805000000],  
    [17.0, 1535808600000]],  
    "target": "Opened"},  
  {"datapoints": [  
    [3.0, 1535790600000],  
    [7.0, 1535794200000],  
    [4.0, 1535797800000],  
    [10.0, 1535801400000],  
    [10.0, 1535805000000],  
    [2.0, 1535808600000]],  
    "target": "Assigned"},  
  {"datapoints": [  
    [3.0, 1535790600000],  
    [5.0, 1535794200000],  
    [10.0, 1535797800000],  
    [3.0, 1535801400000],  
    [5.0, 1535805000000],  
    [2.0, 1535808600000]],  
    "target": "Acknowledged"},  
  {"datapoints": [  
    [3.0, 1535790600000],  
    [3.0, 1535794200000],  
    [4.0, 1535797800000],  
    [3.0, 1535801400000],  
    [3.0, 1535805000000],  
    [2.0, 1535808600000]],  
    "target": "Unacknowledged"},  
  {"datapoints": [  
    [46.0, 1535790600000],
```

[48.0,1535794200000],
[32.0,1535797800000],
[48.0,1535801400000],
[34.0,1535805000000],

```
[36.0,1535808600000]],  
"target":"Resolved"}
```

```
}
```

[get_StatusSituationStats](#)

getStatusSituationStats

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

Request Argument

Name	Type	Description
status	Array	An array of status ids. This is optional. If not given, it returns the default set of statuses: Opened, Unassigned, Assigned, Acknowledged, Unacknowledged, Resolved.
from	Number	Start of the time range. This is a Unix epoch timestamp in seconds.
to	Number	End of the time range. This is a Unix epoch timestamp in seconds. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	"sum" - for an aggregation of all teams provided. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the status
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ol style="list-style-type: none"> 1. Data point – Number of Situations for each status. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past month • 1 year to 5 years – Returns the number of Situations for the past year • More than 5 years – Returns the number of Situations for the past 5 years

- 1 month to 1 year – Returns the number of Situations for the past week
- More than 1 year – Returns the number of Situations for the past month.

Request Example

A cURL request to retrieve the number of opened and assigned Situations from 15.27pm on Sunday, January 14th until 15.27pm on Monday, 15th January 2018:

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

Response Example

A successful request returns the number of Situations for each status:

```
[{
  "datapoints": [
    [32.0, 1516008478000],
    [54.0, 1516030078000]
    [68.0, 1516030078000]
    [82.0, 1516030078000]
    [88.0, 1516030078000]
  ],
  "target": "Opened"
}, {
  "datapoints": [
    [5.0, 1515947278000],
    [12.0, 1515958078000],
    [25.0, 1515976078000],
    [31.0, 1515994078000],
    [40.0, 1516015678000]
  ],
  "target": "Assigned"
}]
```

[getSystemSituationStats](#)

getSystemSituationStats

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

Request Argument

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

		End of the reporting time range. This is a Unix epoch timestamp in seconds.
		If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	"System"
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ul style="list-style-type: none"> 1. Data point – Number of Situations for each status. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past week • More than 1 year – Returns the number of Situations for the past month.

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, 1513669500000],
        [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](#)

getTeamSituationStats

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

Request Argument

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

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the team .
datapoints	Number array	An array of data points. Each data point is an array in the format [data point, timestamp]:

1. Data point – Number of Situations for each status.
2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds).

The time range you provide determines number of the data points per time unit the request returns. For example:

- Less than 1 week – Returns the number of Situations for the past hour
- 1 week to 1 month – Returns the number of Situations for the past day
- 1 month to 1 year – Returns the number of Situations for the past week
- More than 1 year – Returns the number of Situations for the past month.

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]]}
```

[getTopServiceSituationStats](#)

getTopServiceSituationStats

A GET request that returns the number of active Situations impacting a top service over a range of time. Top services are the services that have the most situations impacting them.

Request Argument

Name	Type	Description
from	Number	Start of the reporting time range. This is a Unix epoch timestamp in seconds.
to	Number	End of the reporting time range. This is a Unix epoch timestamp in seconds. If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	"sum" - for an aggregation of all teams provided. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the service
datapoints	Number array	An array of data points. Each data point is an array in the format [data point, timestamp]: <ol style="list-style-type: none"> 1. Data point – Number of Situations for each status. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). The time range you provide determines number of the data points per time unit the request returns. For example: <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past week • More than 1 year – Returns the number of Situations for the past month.

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": "Cloud Service"  
}, {  
    "datapoints": [  
        [7.0, 1538133600000],  
        [3.0, 1538133600000],  
        [6.0, 1538133600000],  
        [14.0, 1538133600000],  
        [9.0, 1538133600000],  
        [8.0, 1538133600000],  
        [12.0, 1538133600000],  
        [8.0, 1538133600000],  
        [4.0, 1538133600000],  
        [6.0, 1538133600000],  
        [3.0, 1538133600000]  
    ]  
}]
```

↳ [getTopTeamSituationStats](#)

getTopTeamSituationStats

A GET request that returns the number of active Situations assign to top teams over a given range of time. Top teams are those teams with the highest number of assigned Situations.

Request Argument

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

		If this timestamp is within 10 seconds of the current system time, the last datapoint returned is the current state datapoint.
aggregation	String	"sum" - for an aggregation of all teams provided. "none" - for no aggregation of results.

Response

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

Successful requests return a JSON object containing the following:

Name	Type	Description
target	String	The name of the team.
datapoints	Number array	<p>An array of data points. Each data point is an array in the format [data point, timestamp]:</p> <ul style="list-style-type: none"> 1. Data point – Number of Situations for each status. 2. Timestamp – Calculation time (UNIX epoch timestamp in milliseconds). <p>The time range you provide determines number of the data points per time unit the request returns. For example:</p> <ul style="list-style-type: none"> • Less than 1 week – Returns the number of Situations for the past hour • 1 week to 1 month – Returns the number of Situations for the past day • 1 month to 1 year – Returns the number of Situations for the past week • More than 1 year – Returns the number of Situations for the past month.

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"
}
]

```

Historic Database

Cisco Crosswork Situation Manager employs two databases, an active database and a historic database to enhance performance of the UI and extend data retention capabilities.

If you are upgrading from Cisco Crosswork Situation Manager v. 6.4 or earlier, manually split the database if you want to benefit from using a separate historic database. See [Enable the Historic Database](#) for further details.

See [Historic Database Benefits](#) for information on the performance and scalability advantages of separating the active and historic databases.

You can use various closing strategies to help maintain your active database at an optimal size:

- Manually close alerts
- Programmatically close alerts
- Use the [Auto Close](#) feature

The historic database can grow without affecting performance. When it is time to retire data from the historic database, you can [archive selected Situations and alerts](#).

Control Historic Data Retention

Historic data retention requires the [Housekeeper Moolet](#) to work. Verify you have configured the Housekeeper Moolet and that it is enabled within moogfarmd. The Housekeeper periodically identifies eligible closed alerts and Situations in the active database and moves them into the historic database.

You can control historic data retention using the database split enabler at \$MOOGSOFT_HOME/bin/utils/moog_db_split_enabler.

See the [Historic Data Utility Command Reference](#) for a full list of available arguments.

Access Historic Data

By default, the database split enabler creates a database called historic_moogdb within the same MySQL instance as the active database: moogdb.

The historic database contains alert and Situation related tables that have the same structure as their equivalents in the active database.

You can access this historic data in the UI in read-only format:

- Search results (when "include close" is selected)
- Direct URL (for Situation Room, Alert Timeline etc.)

- Situation Room (including Situation Alert View and Situation Timeline)
- Similar Situations (historical closed Situations will feature in the Similar Situations list for a Situation)
- PRC feedback can be set for closed alerts from the historic database (as accessed from the Alerts Tab of a Situation Room)

Historic data is also accessible from various [Graze](#) endpoints and their equivalent MoogDb methods:

- `getAlertIds`
- `getAlertDetails`
- `getSituationAlertIds`
- `getSituationDetails`
- `getSituationHosts`
- `getSituationProcesses`
- `getSituationServices`
- `getSituationActions`

Cisco Crosswork Situation Manager rejects Graze endpoints and MoogDb methods that attempt to modify historic alerts and Situations.

Disable Historic Data Retention

When you disable the historic database, the data already in the historic database remains there. The Housekeeper Moolet does not move the data back to the active database. The old historic database is only accessible via SQL queries.

To disable the retention of historic data in a separate database, run the following command:

```
moog_db_split_enabler -d
```

Enable the Historic Database

If you are upgrading from Cisco Crosswork Situation Manager v. 6.4 or earlier you need to manually split the data into an active database and a separate historic database.

Before you split the database, ensure you have met the following requirements:

- The Housekeeper Moolet is configured and running within moogfarmd
- You have the username and password for a MySQL user with schema creation privileges

To split historic data into a separate database, run this command:

```
moog_db_split_enabler -e
```

Alert and Situation data that meet the default criteria are moved. If you do not specify alternative criteria, all closed alerts and Situations that have not been updated within the past hour are moved into a historic database.

See the [Historic Data Utility Command Reference](#) for a full list of default and available arguments. Any changes to the default settings are picked up and applied immediately to a moogfarmd with Housekeeper Moolet running.

If the process encounters closed alerts that are eligible to be moved but still have a relationship with an open Situation, it copies the alerts to the historic database instead of moving them. Later, after they're no longer related to an open Situation, the Housekeeper Moolet moves them to the historic database.

If your system contains a large amount of closed data, the first run may take a long time and require additional CPU and memory use for the moog_farmd process.

When the splitting process is active, you can monitor its progress in moogfarmd.log by running this command:

```
tail -f /var/log/moogsoft/moogfarmd.log|grep Splitter
```

After the database is split, UI based filters do not return closed alerts or Situations that have been moved to the historic database.

Database Split Examples

To split the database with a grace_period and a run_interval of 60 seconds and an alerts_batch_size and a sigs_batch_size of 1000, run this command:

```
moog_db_split_enabler -e -g 60 -r 60 -a 1000 -s 1000
```

During the splitting process you can see entries such as the following in moogfarmd.log:

```
WARN : [0:House][20180315 15:58:44.207 +0000] [CSplitterService.java]:143  
+|Data Splitter started|+  
WARN : [0:House][20180315 15:58:44.503 +0000] [CSplitterTask.java]:205  
+|Splitter will copy [17] alerts and will move [1983] alerts and [500] situations|+  
WARN : [0:House][20180315 15:58:44.706 +0000] [CSplitterTask.java]:205  
+|Splitter will copy [152] alerts and will move [1848] alerts and [0] situations|+  
WARN : [0:House][20180315 15:58:46.434 +0000] [CSplitterTask.java]:205  
+|Splitter will copy [78] alerts and will move [917] alerts and [0] situations|+  
WARN : [0:House][20180315 15:58:47.280 +0000] [CSplitterTask.java]:201  
+|Nothing more to split|+  
WARN : [0:House][20180315 15:58:47.282 +0000] [CSplitterService.java]:145  
+|Data Splitter completed|+
```

If there is no eligible data to move to the historic database, the following entry is logged:

```
WARN : [0:House][20180315      15:12:22.547    +0000] [CSplitterService.java]:143  
+|Data Splitter started|+  
WARN : [0:House][20180315      15:12:22.666    +0000] [CSplitterTask.java]:201  
+|Nothing more to split|+  
WARN : [0:House][20180315 15:12:22.667 +0000] [CSplitterService.java]:145  
+|Data Splitter completed|+
```

Example Split Scenario

The following table illustrates how the Housekeeper Moolet splits an example set of Situations and alerts.

Pre-Split Active Database		Post-Split Active Database	Post-Split Historic Database
Situation 1 (closed) with member alerts: <ul style="list-style-type: none">• Alert 1 (closed)• Alert 2 (closed)• Alert 3 (closed) Situation 2 (closed) with member alerts: <ul style="list-style-type: none">• Alert 4 (closed)• Alert 5 (closed)• Alert 6 (closed) Situation 3 (open) with member alerts: <ul style="list-style-type: none">• Alert 5 (closed)• Alert 6 (closed)• Alert 7 (closed)	Split occurs	Situation 3 (open) with member alerts: <ul style="list-style-type: none">• Alert 5 (closed)• Alert 6 (closed)• Alert 7 (closed)• Alert 8 (open) Situation 4 (Open) with member alerts: <ul style="list-style-type: none">• Alert 8 (open)• Alert 9 (open) Loose alerts: <ul style="list-style-type: none">• Alert 11 (open)	Situation 1 (closed) with member alerts: <ul style="list-style-type: none">• Alert 1 (closed)• Alert 2 (closed)• Alert 3 (closed) Situation 2 (Closed) with member alerts: <ul style="list-style-type: none">• Alert 4 (closed)• Alert 5 (closed)• Alert 6 (closed) Loose alerts: <ul style="list-style-type: none">• Alert 10 (closed) Other alerts:

<ul style="list-style-type: none"> • Alert 8 (open) <p>Situation 4 (open) with member alerts:</p> <ul style="list-style-type: none"> • Alert 8 (open) • Alert 9 (open) <p>Loose alerts:</p> <ul style="list-style-type: none"> • Alert 10 (closed) • Alert 11 (open) 		<ul style="list-style-type: none"> • Alert 7 (closed)
---	--	--

Notes on the above:

- The process copies closed alerts 5 and 6 to the historic database because they are related to open Situation 3. In the historic database they retain their relationship to closed Situation 2, but not open Situation 3 until it is closed and the Housekeeper Moolet moves it to the historic database.
- The process copies closed Alert 7 to the historic database, but within that database the relationship to open Situation 3 is removed, until Situation 3 is closed and the Housekeeper Moolet moves it to the historic database.

Historic Database Benefits

Prior to the release of Cisco Crosswork Situation Manager 6.4, the single-database architecture limited the amount of data you could retain while running a performant system. Systems with tens of millions of alerts could be slow to display the left navigation filters, especially alert filters. Sluggish performance could also affect the rendering of filter data or a refresh after a filter modification. The threshold for acceptable performance hovered around 15M alerts. In very large systems this typically represented two months of data. Aggressive archive management was the only solution to improve performance.

With Cisco Crosswork Situation Manager 6.5, separate active and historic databases are created as part of the installation process. The Housekeeper Moolet periodically migrates closed Situation and alert data from the active to the historic database.

The segmentation of open Situations and alerts from closed ones yields a number of performance and scalability improvements. The extent of the benefits depends on your system size and the size of your database.

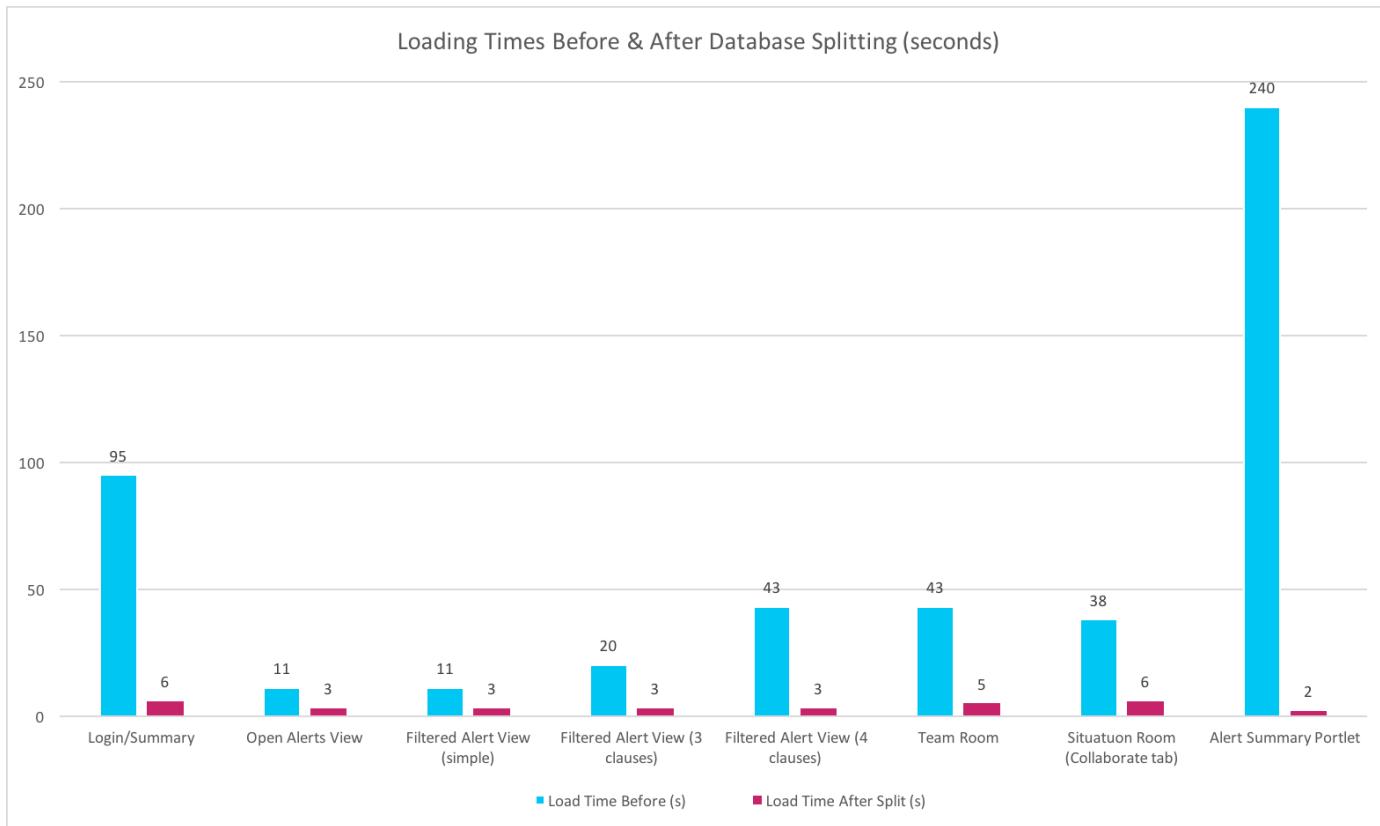
If you are upgrading from Cisco Crosswork Situation Manager 6.4 or an earlier release, manually split the database if you want to benefit from using a separate historic database. See [Historic Database](#) for further details.

Performance Improvement Metrics

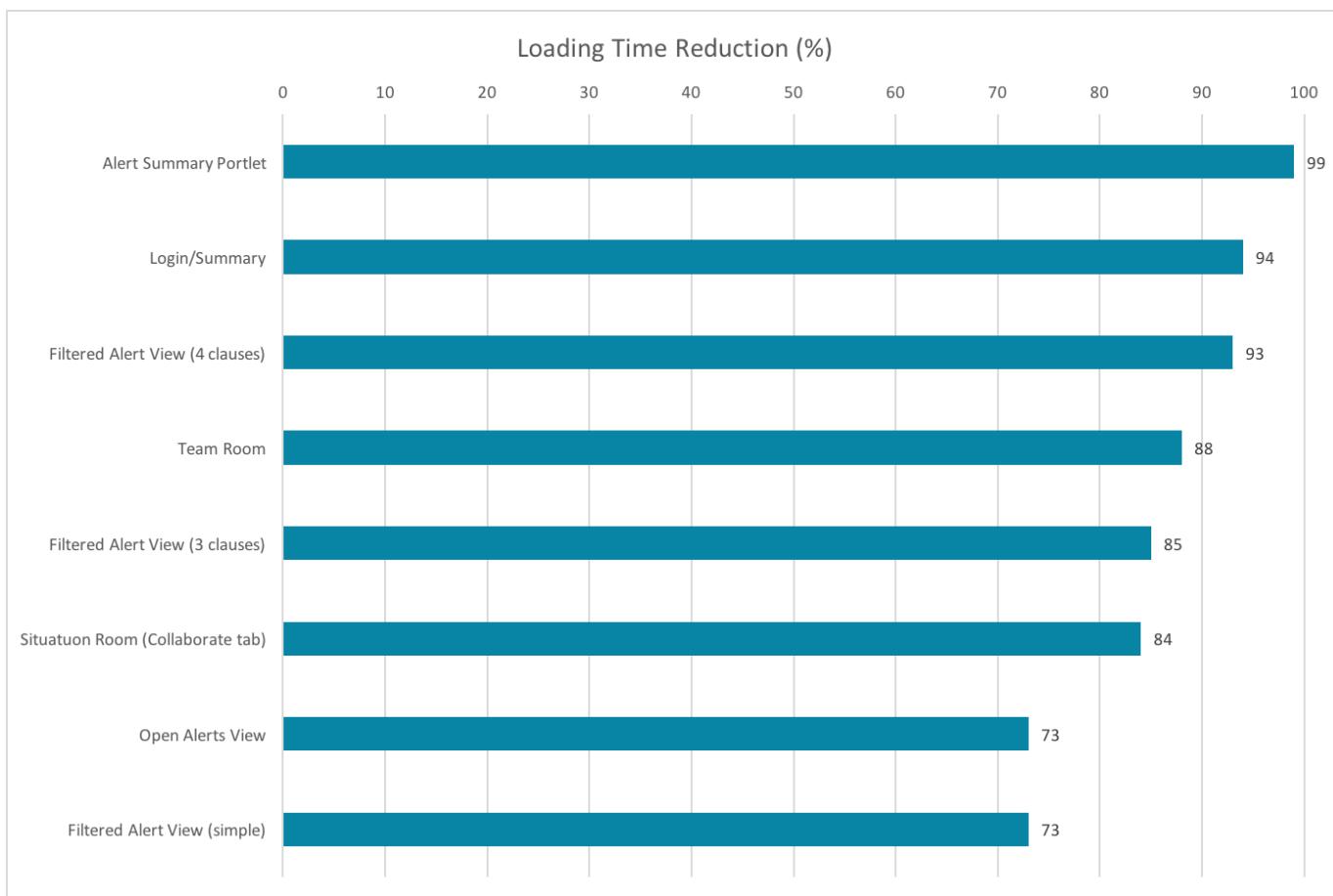
Cisco engineering used the following baselines for testing performance differences between split-database and single-database systems: Single

- Linux server with 64 x 2.3GHz cores
- 128 GB RAM
- 13 months of data from a very large system (100M events, 14M alerts and 1M Situations)
- 3% open alerts and Situations.

Splitting the database improved all filtering and loading times. The bar chart below displays the loading times before and after the database split on 13 months of data. Note: most customers could not have operated in production for 13 months with such performance.



The percentage improvements to loading times for different areas of Cisco Crosswork Situation Manager on the 13 months of data were as follows:



Other improvements included:

- 12% reduction in the time to run a full search re-index
- 12% reduction in the time taken for the Alert Builder to process 1M raw events from a Socket LAM to 600,000 events and 4,000 alerts in the database
- the Cookbook Sigaliser algorithms improved performance.

In this test environment example, the open:closed ratio for alerts and Situations was 3:97. The database split only impacts closed alerts and situations, so you might see different results if your system has a higher proportion of open alerts and Situations.

Next Steps

If you are upgrading to Cisco Crosswork Situation Manager v. 6.5 and your system has performance issues related to the amount of data in your database, you can learn more about the [Historic Database](#) feature and evaluate its benefits:

- Read the documentation: [Historic Database](#)
- Follow the instructions to implement a database split in a non-production environment so you can test the performance improvements and understand any potential impacts to your workflow.
- If you had previously implemented aggressive archiving, relax it incrementally and observe the system and user performance over time.

Be aware of the following impacts when you implement database splitting:

- General system performance depends on the database splitting settings. Aggressive splitting can lead to an increased CPU load because both MySQL and moogfarmd are consuming CPU. You can run the Housekeeper Moolet in its own moogfarmd on a separate host to potentially mitigate any performance impacts.
- Retaining more data means that your database will increase in size. Implement monitors to check your database growth.

Historic Data Utility Command Reference

This is a reference for the [Historic Database](#). The moog-db-split-enabler command line utility accepts the following arguments:

Argument	Input	Description
-a, --alerts_batch_size <arg>	Integer: <number of alerts>	Number of alerts per batch when moving data to the historic database. Increasing this value can speed up the process but places more load on the database. Defaults to 2000.
-d, --disable	-	Disable the retention of historic data in a separate database. The data in the old historic database is not moved to the active database and is effectively inaccessible.
-e, --enable <arg>	String	Enable the retention of historic data in a separate database.
-g, --grace_period <arg>	Integer: <number of seconds>	Period of time since the last update after which closed alerts and Situations are eligible to be included in historic data retention. Defaults to 3600 (one hour).
-h, --help	-	Display the syntax and option descriptions.
-l, --loglevel <arg>	DEBUG INFO WARN FATAL	Specify the output verbosity. Defaults to WARN.
-n,--database_name <arg>	String	Name of the database to contain the historic data. This database is created the first time the utility runs. If the name is changed from a previous run, the data in the old historic database is not moved to the new historic database and is inaccessible from the UI. Defaults to historic_moogdb.
-p,--password <arg>	String	Password for the MySQL user specified in the user argument -u. Defaults to ''.
-r,--run_interval <arg>	Integer: <number of seconds>	The intervals at which the utility runs and moves any eligible data to the historic

		database. The first execution is always the run_interval time divided by two. Defaults to 600 (ten minutes).
-s,--sigs_batch_size <arg>	Integer: <number of Situations>	Number of Situations to include in each batch. Defaults to 500.
-u,--user <arg>	String	MySQL username with schema creation privileges. Defaults to root.

Import a Network Topology

Cisco Crosswork Situation Manager can use Vertex Entropy and Cookbook calculations to group alerts based on their proximity or topological importance. You can use the topology builder utility to import your network topology into Cisco Crosswork Situation Manager so it can run these calculations.

To use topology builder, you create a comma-separated value (.csv) file of the node-to-node connections in your network. The utility builds and caches the topology in the moogdb and moog_reference databases. You can also add optional weight to indicate the value of each edge in the network.

Before You Begin

Before you import your network topology into Cisco Crosswork Situation Manager, ensure you have met the

- following requirements: You have generated a map of the connected nodes in your network in a .csv file.
- Your .csv file contains all of the nodes that are expected to send events.
- The lines in your .csv file follow the format: <node1>,<node2>,<weight>. For example:

```
host_a3,host_a1,3
host_a3,host_a2,3
host_a4,host_a1,3
```

Topology builder logs and rejects any lines in your file that are not in the correct format. It also ignores any loops such as 'host_x, host_x'. The string node names are case insensitive. The weight values can be decimals.

Build a Topology

Build your network topology in Cisco Crosswork Situation Manager as follows:

- Import the .csv topology file into the databases using the topology builder found at \$MOOGSOFT_HOME/bin:

```
./topology_builder -t <file_name>.csv
```

- You can use -l to define the level of debug output. For all options see [Topology Builder Command Reference](#).
- The topology builder utility uses the source data to build a topology. If there is no pre-existing topology, topology builder records host names in the entity_catalogtable. By default topology builder assigns each node to the 'Network' group and the 'Unix servers' competency. See [Configuration Management Database](#) for more details. The utility records the topological information in the moog_references.e.one_hop_topoand moog_reference.topo_nodestables.

After you have imported your topological data, you can use the values in clustering calculations by Nexus and Cookbook. You can also use the graph analyser utility to calculate the [Vertex Entropy](#) of the nodes in your network.

Rebuild a Topology

You can rebuild the topology using the -<percent> option. If you run topology builder and the percentage of new edges in the topology compared to the existing topology exceeds the <percent> value provided, the topology is rebuilt.

For example, if you want to rebuild the topology if at least 50% of the edges are new, run the following:

```
./topology_builder -t <file_name>.csv -r 50
```

To force a rebuild of the topology, run:

```
./topology_builder -t <file_name>.csv -r 0
```

Topology Builder Command Reference

This is a reference for `topology_builder`. The `topology_builder` command-line utility accepts the following arguments:

Argument	Input	Description
<code>-h,--help</code>	-	Display the <code>topology_builder</code> utility syntax and option descriptions.
<code>-l,--loglevel <arg></code>	DEBUG INFO WARN FATAL	Log level controlling the amount of information that <code>topology_builder</code> logs. Defaults to DEBUG.
<code>-r,--rebuild <arg></code>	Integer <percentage>	Percentage of unprocessed topology before a rebuild. Utility rebuilds topology when the percentage of unlabelled nodes exceeds this value. If no percentage value is entered, the topology is not rebuilt.
<code>-t,--topology-file <csv_filename></code>	String <csv_filename>	Name of the .csv file containing the pairs of connected nodes with an optional weighting value.

Logging

Message Bus

- [Message Handling](#)
- [Default Configuration](#)
- [Zones](#)
- [Message Persistance](#)

The Messaging System (MooMS) is the message bus component of Cisco Crosswork Situation Manager and shares event data. This is subscribed to by the various Moolets.

The message bus is a publish-subscribe message brokering system implemented with [RabbitMQ](#) which uses [AMQP](#), an open standard for message-orientated middleware, over TCP.

Message Handling

The Message Bus handles the data it receives (e.g. raw event data, new alerts, Situation activity etc) by placing it in queues, which are lines of messages waiting to be handled.

Cisco Crosswork Situation Manager does not enforce any size or time limits on queues, so the maximum number of messages in a queue is limited by the available RAM and disk space on the server. It also depends on the size of the Alerts and Situations being generated. The size limit is 128kb for Alerts and 64kb for Situations.

Once the maximum number of messages has been reached, the broker drops messages from the front of the queue to make room for new messages. By default, Cisco Crosswork Situation Manager applications use exclusive transient queues. For example, if Cisco Crosswork Situation Manager or the broker shuts down or dies then the queue and all of its messages will be lost. Durable queues can be enabled using the `message_persistence` setting in `$MOOGSOFT_HOME/config/system.conf` (see [Message Persistance](#)).

For more information see the RabbitMQ docs on [queues](#) and [queue length](#).

Default Configuration

By default, Cisco Crosswork Situation Manager installs with a single RabbitMQ broker running on the same machine as the other Cisco Crosswork Situation Manager components (LAMs, moog_farmd, the Moolets, etc.).

The out-of-the-box configuration is:

```
port: 5672 zone:  
<none>  
username: moogsoft password:  
m00gs0ft.
```

The username and password always need to match the Message Bus broker configuration. If commented out, a default "guest" user will be used (guest: guest).

Zones

You can use zones, or virtual hosts, to share a single RabbitMQ broker cluster among multiple instances of Cisco Crosswork Situation Manager.

If multiple instances of Cisco Crosswork Situation Manager share a single RabbitMQ broker then each instance uses a different zone name to prevent message interference. In RabbitMQ, a zone is called a virtual host (vhost). Clients connecting to one vhost cannot see messages sent to a different vhost.

The default deployment does not use zones. If you specify a zone name, you must also configure a vhost with the same name in the RabbitMQ broker.

By default, Cisco Crosswork Situation Manager clients connect to the vhost specified during moog-init.sh setup with the "moogsoft" username and the password.

For distributed installations using multiple RabbitMQ brokers, this must be configured. A zone (vhost) name is required by the moog-init.sh setup script. See [Message System Deployment](#).

Message Persistence

You can control and minimize message loss during a shutdown or failure using the following settings in system.conf:

message_persistence

Controls whether MooMS will persist important messages or not in the event of an application or message broker restart. This affects how the message bus handles messages.

Type: Boolean

Default: false

max_retries

The number of attempts to re-send a failed message, used in conjunction with retry_interval.

Type: Number
Default: 100

retry_interval

The time to wait in milliseconds between each re-send attempt. The combination of 100 retries and a 200 msec retry_interval gives a total of 20 seconds, which is typical duration for broker failover in a high availability environment.

Type: Number
Default: 200

cache_on_failure

Controls whether the message is cached internally and resent. If enabled, a message is cached if an initial retry, controlled by max_retry and retry_interval, is a failure.

Type: Boolean
Default: false

cache_ttl

Specifies how many seconds a cached message lives for in the cache list. The system will attempt to resend any cached messages in the order they were put on the cache until their cache time-to-live (cache_ttl) value has been reached. Any messages not successfully sent will be discarded. This value will have a direct impact on sender process memory.

Type: Number
Default: 900

confirmation_timeout

Defines the time to wait for confirmation from a RabbitMQ Broker that it has received the sent message.

Moogsoft advises you do not change this value.

Type: Number

Default: 2000

Message System Deployment

The Message Bus is the message system for Cisco Crosswork Situation Manager, implemented with RabbitMQ. By default, Cisco Crosswork Situation Manager installs with a single RabbitMQ broker running on the same server as the other Cisco Crosswork Situation Manager components (LAMs, moofarmd, the Moolets, etc.). You can also configure Moogsoft Cisco Crosswork Situation Manager as a distributed system with multiple RabbitMQ broker hosts.

If you have any trouble with your deployment see [Troubleshooting](#).

Distributed Deployment

Depending on the systems you monitor, you can increase the performance and reliability of your Cisco Crosswork Situation Manager deployment with distributed RabbitMQ brokers running on different hosts. Execute the following procedure on each RabbitMQ broker host to install a distributed messaging system:

1. Install the Erlang repository:

```
wget http://packages.erlang-solutions.com/erlang-solutions-1.0-1.noarch.rpm  
rpm -Uvh erlang-solutions-1.0-1.noarch.rpm
```

2. Install RabbitMQ Server:

```
rpm --import https://www.rabbitmq.com/rabbitmq-signing-key-public.asc yum install rabbitmq-server-  
3.5.4-1.noarch.rpm
```

3. Copy rabbitmq.config from \$MOOGSOFT_HOME/etc/cots/rabbitmq/rabbitmq.config and add it to the following location on each RabbitMQ broker host:

```
/etc/rabbitmq/rabbitmq.config
```

4. Run the following commands:

```
chkconfig rabbitmq-server on service  
rabbitmq-server start
```

5. Create a new zone (a RabbitMQ "vhost") on each remote RabbitMQ broker and set the permissions for the default user:

```
rabbitmqctl add_vhost <zone>  
Creating vhost "<zone>" ...  
  
rabbitmqctl add_user moogsoft <password>  
Creating user "moogsoft" ...
```

```
rabbitmqctl set_permissions -p <zone> moogsoft ".*" ".*" ".*" Setting permissions for user  
"moogsoft" in vhost "<zone>" ...
```

6. Edit the "mooms" section in system.conf on the Cisco Crosswork Situation Manager host system, to point to the correct IP addresses and ports (two specified in the example below):

```
"mooms" :  
{  
    "zone" : "<zone>",  
    "brokers" : [  
        {  
            "host" : "172.16.87.131",  
            "port" : 5678  
        },  
  
        {  
            "host" : "172.16.87.135",  
            "port" : 5672  
        }  
    ],  
    "username" : "moogsoft",  
    "password" : "<password>"  
},
```

Cluster Message Bus Brokers

The Message Bus broker is a logical grouping containing one or more Erlang nodes each running RabbitMQ and sharing vhosts, users, queues etc. If you have multiple RabbitMQ brokers running, you should cluster them.

For more information about broker clustering, refer to the RabbitMQ documentation on [clustering](#).

Enable Queue Mirroring

As part of a Cisco Crosswork Situation Manager High Availability (HA) deployment that employs message persistence, you must set up mirroring for the relevant durable queues across all nodes in a RabbitMQ cluster.

To enable queue mirroring, run the following command from any host running a broker in the RabbitMQ cluster:

```
rabbitmqctl set_policy -p <zone> ha-all ".+\.HA" '{"ha-mode":"all"}'
```

For the <zone>, specify the zone you used when you initialized your system. For example:

```
rabbitmqctl set_policy -p MoogsoftAIOps ha-all ".+\.HA" '{"ha-mode":"all"}'
```

This command configures mirroring for all the *.HA queues across all RabbitMQ brokers in the cluster.

Run the following command from any host running a broker in the RabbitMQ cluster to verify the policy is enabled:

```
rabbitmqctl -p <zone> list_policies
```

For example:

```
rabbitmqctl -p MoogsoftAIOps list_policies Listing policies for
vhost "MoogsoftAIOps" ...
MoogsoftAIOps ha-all .+\.HA all {"ha-mode":"all"} 0
```

For more information about queue mirroring, refer to the [RabbitMQ docs](#).

Message System Troubleshooting

- RabbitMQ Broker Fails to Start
- Typical Error Messages
- First startup of RabbitMQ broker
- LAMs fail to start from command line
- Manually configure IP address and port
- Manually set up a user or zone (vhost)

The message bus or MooMS is the message system for Cisco Crosswork Situation Manager, implemented with RabbitMQ.

This guide outlines some common issues with the message bus deployment and offers alternative solutions.

Open the Management Console

You can launch the RabbitMQ management console directly from the Cisco Crosswork Situation Manager UI. This provides useful statistics when debugging.

To open the console, go to **Settings > Self Monitoring > Message Bus** and click **Launch Message Bus Console ...** You can log in using the default credentials:

Username: moogsoft
Password: m00gs0ft

These are defined in system.conf. If commented out, a default 'guest' user can be used.

Examine the Log Files

Troubleshooting your MooMS deployment often requires examining log files. The default locations of log files are as follows:

- moog_farmd and LAMs - /usr/log/moogsoft/\$SERVICE_NAME.log where \$SERVICE_NAME is the process, for example socketl amd for the Socket LAM
- Tomcat -/usr/share/apache-tomcat/logs/catalina.out RabbitMQ -
- \$RABBITMQ_HOME/var/log/rabbitmq

RabbitMQ Broker Fails to Start

If RabbitMQ broker fails to start and Security-Enhanced Linux ([SELinux](#)) is enabled, it may be related to this. SELinux and similar mechanisms such as firewalls may prevent RabbitMQ from binding to a port and starting up.

If SELinux is enabled, check that the following ports can be opened:

- 4369 (Erlang port mapper daemon)
- 25672 (Erlang distribution)
- 5672, 5671 (AMQP 0-9-1 without and with TLS)
- 15672 (if management plugin is enabled)

You may need to configure RabbitMQ to use different ports.

For more information, refer to the RabbitMQ install page: <https://www.rabbitmq.com/install-rpm.html>

If the RabbitMQ broker fails to start it may be due to file permissions, you may see errors such as:

```
{error_logger,{ {2015,9,1},{21,26,14} },"Failed to create cookie file '/home
/moogsoft/.erlang.cookie': eacces".[]}

{error_logger,{ {2015,9,1},{21,26,14} },crash_report,[ [{initial_call,{auth,init,['Argument_1']}},{pid,<0.21.0>},{registered_name,[]},{error_info,
{exit,{ "Failed to create cookie file '/home/moogsoft/.erlang.cookie'"}}}]}
```

```
eacces",[{auth,init_cookie,0,[{file,"auth.erl"},{line,286}]},{auth,init,1,[{file,"auth.erl"},{line,140}]},{gen_server,init_it,6,[{file,"gen_server.erl"},{line,328}]},{proc_lib,init_p_do_apply,3,[{file,"proc_lib.erl"},{line,239}]},{gen_server,init_it,6,[{file,"gen_server.erl"},{line,352}]},{proc_lib,init_p_do_apply,3,[{file,"proc_lib.erl"},{line,239}]},{ancestors,[net_sup,kernel_sup,<0.10.0>]},{messages,[]},{links,[<0.19.0>]},{{dictionary,[]},{trap_exit,true}},{status,running},{heap_size,610},{stack_size,27},{reductions,975}],[]}...]
```

When the RabbitMQ broker is running as a service, use the following command to check that it is running:

```
service rabbitmq-server status
```

Typical Error Messages

The section below will outline examples and solutions to typical error messages with RabbitMQ.

Connection refused/ Unable to create RabbitMQ connection

If the RabbitMQ broker appears to be down or unreachable, trying to start an Cisco Crosswork Situation Manager component gives warnings such as:

```
WARN : [main ][20150812 16:09:54.792 +0100][CMoomsFactory.java]:707  
+|Unable to create RabbitMQ connection : [amqp://localhost:5672/ZONE]|+ WARN : [main ][20150812  
16:09:54.792 +0100][CMoomsFactory.java]:256  
+|Unable to create RabbitMQ connection : [java.net.ConnectException: Connection refused]|+  
WARN : [main ][20150812 16:09:54.793 +0100] [CMoomsFactory.java]:707  
+|Unable to create RabbitMQ connection : [amqp://localhost:5672/ZONE]|+ WARN : [main ][20150812  
16:09:54.793 +0100][CJNIMoomsWrapper.java]:253  
+|Problem during mooms setup, retrying|+
```

The structure of the amqp url is: amqp://<hostname>:<port>/<zone>

Solution:

Check if the RabbitMQ broker is running:

```
service rabbitmq-server status
```

If it isn't running, see [RabbitMQ broker fails to start](#) (above).

If it is running, check the MooMS configuration in the 'mooms' section of system.conf.

AlreadyClosedException/broker forced connection closure

If Cisco Crosswork Situation Manager components are running, the RabbitMQ broker going down or becoming unreachable gives warnings such:

WARN : [Thread-][20150812 16:15:42.295 +0100] [CLogger.java]:337 +|Problem sending message id : [4115e512-aa63-44d5-bdc9-8ed164cd75e5] com.rabbitmq.client.AlreadyClosedException: connection is already closed due to connection error; protocol method: #method<connection.close>(reply- code=320, reply-text=CONNECTION_FORCED - broker forced connection closure with reason 'shutdown', class-id=0, method-id=0)

```
at com.rabbitmq.client.impl.AMQChannel.ensureIsOpen(AMQChannel.java:195) at  
com.rabbitmq.client.impl.AMQChannel.transmit(AMQChannel.java:309)  
at com.rabbitmq.client.impl.ChannelN.basicPublish(ChannelN.java:657) at  
com.rabbitmq.client.impl.ChannelN.basicPublish(ChannelN.java:640) at  
com.rabbitmq.client.impl.ChannelN.basicPublish(ChannelN.java:631)  
at com.rabbitmq.client.impl.recovery.AutorecoveringChannel.basicPublish  
(AutorecoveringChannel.java:168)  
at com.moogsoft.mooms.CMoomsMessageSender.send(CMoomsMessageSender.java: 530)  
at com.moogsoft.mooms.CMoomsMessageSender.send(CMoomsMessageSender.java: 448)  
at com.moogsoft.mooms.CMoomsMessageSender.send(CMoomsMessageSender.java: 264)  
at com.moogsoft.mooms.CMoomsMessageSenderPool.send(CMoomsMessageSenderPool.java:378)  
at com.moogsoft.mooms.CJNIMoomsWrapper.sendEvent(CJNIMoomsWrapper.java:288)  
|+
```

If Cisco Crosswork Situation Manager is configured to connect to RabbitMQ brokers in a high availability environment, during a RabbitMQ broker fail-over, warning messages may be logged for a short time

Solution:

If the problem is not temporary, check if the RabbitMQ broker is running:

```
service rabbitmq-server status
```

Problem during mooms setup, retrying

A Cisco Crosswork Situation Manager component trying to connect to a non-existent zone (vhost) in a RabbitMQ broker gives warnings such as:

```
DEBUG: [main ][20150812 16:24:00.764 +0100] [CMoomsFactory.java]:206  
+|Setting factory zone to : [fish]|+  
WARN : [main ][20150812 16:24:03.825 +0100][CMoomsFactory.java]:707  
+|Unable to create RabbitMQ connection : [amqp://localhost:5672/fish]|+ WARN : [main ][20150812  
16:24:03.825 +0100][CMoomsFactory.java]:256  
+|Unable to create RabbitMQ connection : [java.io.IOException]|+ WARN : [main ][20150812  
16:24:06.373 +0100][CMoomsFactory.java]:707  
+|Unable to create RabbitMQ connection : [amqp://localhost:5672/fish]|+ WARN : [main ][20150812  
16:24:06.373 +0100][CJNIMoomsWrapper.java]:253  
+|Problem during mooms setup, retrying|+
```

Solution:

Check you have the correct zone configured in the 'mooms' section of system.conf and that the zone (vhost) has been created in the RabbitMQ broker.

The best way to check that the zone (vhost) has been created in the RabbitMQ broker is to use the RabbitMQ management console.
If the zone (vhost) has not been created, [manually create it via the command line](#), or via the RabbitMQ Management console.
Once the zone has been created, the user defined in system.conf must be given permissions to access the new zone.

AuthenticationFailureException: ACCESS_REFUSED

A Cisco Crosswork Situation Manager component trying to connect to a valid zone (vhost) in a RabbitMQ broker, but with wrong authentication details gives warnings such as:

```
WARN : [main ][20150812 16:20:29.760 +0100] [CMoomsFactory.java]:707
+|Unable to create RabbitMQ connection : [amqp://jimmy@localhost:5672/null]
|
WARN : [main ][20150812 16:20:29.760 +0100][CMoomsFactory.java]:256
+|Unable to create RabbitMQ connection : [com.rabbitmq.client. AuthenticationFailureException:
ACCESS_REFUSED - Login was refused using authentication mechanism PLAIN. For details see the
broker logfile.]|+ WARN : [main ][20150812 16:20:29.805 +0100] [CMoomsFactory.java]:707
+|Unable to create RabbitMQ connection : [amqp://jimmy@localhost:5672/null]
|
WARN : [main ][20150812 16:20:29.805 +0100] [CJNIMoomsWrapper.java]:253
+|Problem during mooms setup, retrying|+
```

Solution:

Check you have the correct username and password in the 'mooms' section of system.conf and that they match those defined in the RabbitMQ broker. If they are correct in system.conf then you must correct it in the RabbitMQ broker. Do this either [via the command line](#), or via the RabbitMQ management console.

Also see [RabbitMQ broker fails to start](#) (above).

First startup of RabbitMQ broker

The first time a RabbitMQ broker is started, it creates an 'account' with the default user and password from rabbitmq.config. If this information is subsequently edited in rabbitmq.config, and the RabbitMQ broker is restarted, the 'account' is not created, which can be confusing.

If the RabbitMQ broker has been started before, then the 'account' will need to be added manually (see [Manually set up a user](#)) rather than by defining a default user in the rabbitmq.config file.

LAMs fail to start from command line

If LAMs run from the command line or as a service result in the following error:

```
[root@moogbox2 bin]# ./socket_lam
./socket_lam: error while loading shared libraries: libjvm.so: cannot open shared object file: No such file or
directory
```

...it may be because /usr/java/jdk1.8.0_20/jre/lib/amd64/server has not been added to the LD_LIBRARY_PATH.

- To run the LAMs via a command line, a change the LD_LIBRARY_PATH to be as follows (the default initfiles have this modification made):

```
export LD_LIBRARY_PATH=$MOOGSOFT_HOME/lib:/usr/GNUstep/Local/Library
/Libraries:/usr/GNUstep/System/Library/Libraries:$JAVA_HOME/jre/lib/amd64
/server
```

Manually configure IP address and port

In some environments (such as SELinux) you may need to configure a RabbitMQ broker to listen on a different IP address and port. To do this:

- Configure the contents of /etc/rabbitmq/rabbitmq-env.conf, for example:

```
RABBITMQ_NODE_IP_ADDRESS="172.168.87.131
RABBIT_NODE_PORT="5678"
```

2. Restart the rabbitmq-server service:

```
service rabbitmq-server restart
```

Manually set up a user or zone (vhost)

To help troubleshoot an existing RabbitMQ broker, you may want to manually set up a user or zone (vhost).

- To manually set up a new user in the RabbitMQ broker, run the following command (using your own user, password and zone):

```
rabbitmqctl add_user <user> <password>
rabbitmqctl set_permissions -p <zone> <user> ".*" ".*" ".*"
```

Also ensure the username, password and zone in the 'mooms' section of system.confmatch those defined in the RabbitMQ broker with the above commands.

- To manually set up a new zone in the RabbitMQ broker, run the following command (using your own user and zone):

```
rabbitmqctl add_vhost <zone>
rabbitmqctl set_permissions -p <zone> <user> ".*" ".*" ".*"
```

Also ensure the username and zone in the 'mooms' section of system.confmatch those defined in the RabbitMQ broker with the above commands.

Message System SSL

The MooMS message system can be configured to operate using SSL connections to provide secure and authorized connectivity.

MooMS is the message system for Cisco Crosswork Situation Manager, implemented with RabbitMQ. By default, Cisco Crosswork Situation Manager provides rabbitmq.config which does not start RabbitMQ in SSL mode.

To enable RabbitMQ to run in SSL mode, see documentation at: <https://www.rabbitmq.com/ssl.html>

Configure Cisco Crosswork Situation Manager to use SSL with MooMS

Once RabbitMQ has been configured to use SSL, Cisco Crosswork Situation Manager needs to be configured to use the RabbitMQ broker's SSL port, as well as the SSL certificates and keys to enable secure and authorized connection to these brokers if required by the SSL configuration set on RabbitMQ.

Below is an example of full SSL configuration of MooMS in system.conf:

system.conf

```
#####
# SSL configuration can be used to provide a means of secure
# communication between a Moog process and MooMS. MooMS can be setup # with options to
# accept SSL connections with or without providing
# the relevant certificates and keys. #
# Three modes of SSL are available:
#
# 1. No SSL           - SSL configuration is not specified
#
```

```

# 2. Express SSL - This is where SSL configuration is specified, but          #
#                           empty or only the SSL protocol is set and specific      #
#                           certificates do not need to be specified.          #
#
# 3. Custom SSL           - This is where all the SSL configuration and      #
#                           certificates needed are specified to enable secure       #
#                           and authorised communication to MooMS.                 #
#
#                           Note that Client key and certificate are optional.      #
#                           If neither of those are specified, then client           #
#                           certification verification will not be performed. #
#####
,"ssl" :
{
#
# Specify the SSL Protocol to use.
# If the configuration is not specified, "TLSv1.2" will be used # by default.
# JRE 8 supports "TLSv1.2", "TLSv1.1", "TLSv1", "SSLv3" #
"ssl_protocol" : "TLSv1.2",

#
# The location of the SSL certificate, key files. #
# Relative pathing can be used, i.e. '.' to mean current directory, # '../server.pem' or '../../server.pem'
etc. If neither relative
# nor absolute (using '/') path is used then $MOOGSOFT_HOME is # prepended to it.
# i.e. "config/server.pem" becomes "$MOOGSOFT_HOME/config/server.pem" #

#
# Specify the server certificate. #
"server_cert_file" : "server.pem",

#
# Enable client authentication by specifying the client certificate # and key files below.
# The key file has to be in PKCS#8 format. #
"client_cert_file" : "client.pem", "client_key_file"
                      : "client.key"
}

```

Express SSL

Cisco Crosswork Situation Manager can be configured to connect to the RabbitMQ server without validating any certificates or attempting to authorize the client. If the RabbitMQ server has been configured to reject clients that do not present valid certificates then this SSL mode will not work, Cisco Crosswork Situation Manager will need to be configured with the correct certificates and keys to establish connectivity. To enable express SSL mode simply uncomment "ssl" configuration block, optionally specify the "ssl_protocol" configuration:

Express SSL

```
"ssl" :  
{  
    #  
    # Specify the SSL Protocol to use.  
    # If the configuration is not specified, "TLSv1.2" will be used # by default.  
    # JDK 8 supports "TLSv1.2", "TLSv1.1", "TLSv1", "SSLv3" #  
    "ssl_protocol" : "TLSv1.2"  
}
```

Custom SSL

Cisco Crosswork Situation Manager can be configured to connect to the RabbitMQ server using a specific server certificate, and if RabbitMQ has been enabled with Client Authentication then Cisco Crosswork Situation Manager can be configured with the client key and client certificate to authenticate with RabbitMQ.

Client Authentication is optional functionality, to run Cisco Crosswork Situation Manager with just a specific server certificate simply comment out the `client_cert_file` and `client_key_file` entries.

If Client Authentication is used, the "`client_key_file`" must be in a PKCS#8 Format. The following command can be run to convert a private key in to PKCS#8 format:

```
openssl pkcs8 -topk8 -inform PEM -outform PEM -nocrypt -in key.pem - out client.key
```

An example of Cisco Crosswork Situation Manager specifying full SSL configuration, connecting to a RabbitMQ which requires Client Authentication. The example also shows how you can organise the server and client SSL files in sub-folders:

Custom SSL

```
,"ssl" :  
{  
    #  
    # Specify the SSL Protocol to use.  
    # If the configuration is not specified, "TLSv1.2" will be used # by default.  
    # JRE 8 supports "TLSv1.2", "TLSv1.1", "TLSv1", "SSLv3" #  
    "ssl_protocol" : "TLSv1.2",  
  
    #  
    # The location of the SSL certificate, key files. #  
    # Relative pathing can be used, i.e. '.' to mean current directory, # '../server.pem' or '../../server.pem'  
    etc. If neither relative  
    # nor absolute (using '/') path is used then $MOOGSOFT_HOME is # prepended to it.  
    # i.e. "config/server.pem" becomes"$MOOGSOFT_HOME/config/server.pem"
```

```

#
#
# Specify the server certificate. #
"server_cert_file" : "server/server.pem",

#
# Enable client authentication by specifying the client certificate # and key files below.
# The key file has to be in PKCS#8 format. #
"client_cert_file" : "client/client.pem",
"client_key_file"      : "client/client.key"

}

```

To disable SSL connectivity with MooMS, change the port number for the brokers back to the non-SSL port (typically 5672) and comment out the "ssl" section in system.conf

Moog Encryptor

Cisco Crosswork Situation Manager includes an encryptor utility so you can encrypt passwords stored in the system.conf configuration file. Encrypted passwords in configuration files are more secure because someone with access to the configuration cannot necessarily gain access to integrated systems.

If you run in a distributed environment, run the encryptor utility on one host to create an encryption key (.key). Then copy the key to the \$MOOGSOFT_HOME/etc/directory on the remaining hosts.

Encrypt a Password

To encrypt a password, execute the moog_encryptor command as follows:

```
$MOOGSOFT_HOME/bin/moog_encryptor -p <password>
```

For example, to encrypt a password "Abacus":

```
/usr/share/moogsoft/bin/moog_encryptor -p Abacus
```

The moog_encryptor displays the encrypted password:

The encrypted password is:

```
KfFJGilmGGJP/qTrJV6SBs0HTTy3NpCqvGaYKviDbLQ=
```

When using within Javascript code or JSON file, use:

```
{"encrypted_password":"KfFJGilmGGJP
/qTrJV6SBs0HTTy3NpCqvGaYKviDbLQ="}
```

Each time you run moog_encryptor, it generates a different encrypted password.

Configure Cisco Crosswork Situation Manager to use Encrypted Passwords

You can use passwords encrypted with moog_encryptor in the system.conf file as follows:

1. Edit \$MOOGSOFT_HOME/config/system.conf.
2. Identify the password you want to replace and uncomment the encrypted_password property. Comment out the password property. For example:

```
"username" : "moogsoft",
#"password" : "Abacus".
"encrypted_password" : "e5uO0LY3HQJZCltG/caUnVbxVN4hImm4gIOpb4rwpF4=",
```

3. Set the value of the encrypted_password property to the value returned from the moog_encryptor. For instance:

```
"encrypted_password":"KfFJGilmGGJP/qTrJV6SBs0HTTy3NpCqvGaYKviDbLQ=",
```

4. Change the value of the password property so that it does not match the unencrypted value of the password.

Change the Location of the Encryption Key

By default, the encryptor utility uses a key at the following location:

```
$MOOGSOFT_HOME/etc/.key
```

The encryptor utility creates a new key if one does not already exist.

If you want to use a different location for the key, uncomment the encryption section in **system.conf**. Set the value of the encryption_key_file property to a new path for the key. For example:

```
# Uncomment the encryption section if you want to specify the location
# for the encryption key file.

,
"encryption" :
{
    # Use this to change the default location of the encryption
key file
    "encryption_key_file" : "/usr/share/example/.key"
}
#
```

You must configure Cisco Crosswork Situation Manager to use the same .key file you used to encrypt passwords. If you encrypt a password using one key and then change the configuration to use another key, decryption fails.

Search and Indexing

Cisco Crosswork Situation Manager uses [Elasticsearch](#) to provide search and data indexing functions.

You can control the Elasticsearch service using the following service script:

```
/etc/init.d/elasticsearch [start|restart|stop]
```

All Elasticsearch logs are stored in following location:

```
/var/log/elasticsearch/
```

Perform an Index

There are two tools you can use to index Alerts and Situations: the Indexer Moolet and the `moog_indexer` utility.

Indexer Moolet

The Indexer listens for new Alerts and Situations on the Message Bus and indexes them. Cisco Crosswork Situation Manager indexes Alerts and Situations as soon as they are created or changed so you can search for them immediately.

You can configure the Indexer in `moog_farmd.conf` using the following parameters:

`enable_private_teams`

Set to 'True' if you limit team permissions based upon services, Situations, or alerts assigned to the team. The indexer applies team permissions to the indexes.

If disabled, Indexer will index all Alerts and Situations present in Cisco Crosswork Situation Manager.

Type: Boolean

Default: False

`full_scan_batch_size`

The maximum number of Alerts or Situations the Indexer scans in each batch as it indexes. This is useful because it is not possible to load all Alerts to the memory in one go.

By default Indexer scans through batches of 1000 Alerts or Situations.

Type: Integer

Default: 1000

`full_scan_wait`

The number of seconds the Indexer waits between batches. One advantage of this is to free up the CPU and memory being used to index each batch.

It is set to 0 by default so the Indexer will not wait between batches.

Type: Integer

Default: 0

`full_scan_at`

Determines the exact time when Indexer runs a full scan. This allows you to ensure the accuracy of search data once per day by performing a full re-index. If left empty, the Indexer does not perform a full scan.

Type: Time (HH:mm:ss)

Default: "02:12:35"

full_scan_at_startup

If enabled, then Indexer performs a full scan when it starts. This is useful if you are not using the scheduled scan and only restart moogfarmd once a week.

Type: Boolean
Default: false

historic_scan_frequency

Determines how frequently the Indexer performs a full scan of both active and historic databases. By default, Indexer scans both databases every three days.

Type: Integer
Default: 3

For example, refer to the default Indexer configuration:

```
# Set to false to disable private teams indexing. enable_private_teams:  
false,  
  
# Maximal full scan batch size  
full_scan_batch_size: 1000,  
  
# How many seconds to wait between batches (0 not to wait) full_scan_wait: 0,  
  
# When to run the full scan (HH:mm:ss) leave empty to disable full scan (HH:mm:ss)  
full_scan_at: "02:12:35",  
  
# Do we want to run full scan when the moolet starts? full_scan_at_startup:  
false  
  
# Scan the historic data once every how many full scans historic_scan_frequency:  
3
```

moog_indexer

You can run the indexer utility alongside the Indexer Moolet. Before you can run the indexer utility, you must start moogfarmd with a running Indexer Moolet.

The moog_indexer command-line utility accepts the following options:

Argument	Input	Description
-h,--help	-	Displays the help text with arguments that can be used with the utility.
-f, --full	-	Scans both the active and historic data. Use this argument if you want data from both databases to be indexed.
-i,--in <arg>	Integer	Schedule full index to run in a set amount of time (in hours). This can be a decimal. For example, 0.1 = 6 minutes.
-l,--loglevel <arg>	INFO WARN ALL	Specify the log level to choose the amount of debug output
-n,--now	-	Schedules a full index to run immediately.

-r,--report

-

Request report from on the last performed full scan index. This report will show the status of previous runs within the lifetime of the moogfarmd process and any runs still in progress. If moogfarmd is restarted, the -r argument will not return any data.

Please note: If you use Private Teams mode, meaning one or more Roles do NOT have the all_data permission set, then you must run both the initial 'full index' and the 'incremental index crontab' moog_indexer commands with the -p argument. If not, users in one Team will be able to see search results for other Teams.

Tune your MySQL database to ensure indexing runs as quickly as possible. See either the [Percona](#) or [MySQL](#) websites for information on tuning and optimization.

An output example is shown below:

```
[root@myhost home]# moog_indexer -r
Got report:
05/10/17    13:43:06 - Starting full scan
05/10/17    13:43:06 - Scanning for alerts
05/10/17    13:43:07 - Scanned: [177] alerts
05/10/17    13:43:07 - Scanning for situations
05/10/17    13:43:07 - Scanned: [44] situations
05/10/17    13:43:07 - Full scan complete
05/10/17    13:43:22 - Starting full scan
05/10/17    13:43:22 - Scanning for alerts
05/10/17    13:43:22 - Scanned: [204] alerts
05/10/17    13:43:22 - Scanning for situations
05/10/17    13:43:23 - Scanned: [55] situations
05/10/17    13:43:23 - Full scan complete
```

Before you upgrade to Cisco Crosswork Situation Manager V6.2.1 or later from a version older than 6.2.1, remove or disable the crontab jobs for the old indexer utility.

Elasticsearch Details

Elasticsearch runs on port 9200 by default.

To make Elasticsearch available externally and listen on the external host IP address, run the following command:

```
$MOOGSOFT_HOME/bin/utils/moog_init_search.sh -r
```

The script updates the Elasticsearch configuration and restarts the service.

Situation Merge Behavior

Cisco Crosswork Situation Manager uses a configuration called the `sig_similarity_limit` to automatically merge similar Situations. When two Situations reach this similarity limit, Cisco Crosswork Situation Manager merges them.

The default similarity limit for clustering algorithms is '0.7' so Situations sharing 70% of the same Alerts are merged.

Merge Groups

You can use `merge_groups` in `moog_farmd.conf` to control how Cisco Crosswork Situation Manager merges Situations created by different clustering algorithms.

Any Sigaliser/moolet not defined in a new merge group belongs to the default group. By default, Cisco Crosswork Situation Manager merges Situations when they meet the following criteria:

```
alert_threshold : 2,  
sig_similarity_limit : 0.7
```

To override the default behavior, you can create custom merge groups.

Create a Merge Group

You can create merge groups by following these steps:

1. Edit moog_farmd.conf.
2. Define new merge groups in the merge_groups section. For example:

```
# {#  
# #      name: "Merge Group 1",  
#       moolets: ["Nexus", "Tempus"].  
# }      alert_threshold : 3,  
#       sig_similarity_limit : 0.75
```

This merge group would only merge Situations created by the Nexus and Tempus Sigalisers which shared 75% of the same Alerts.

Each new merge group must be given a name and can be defined using the following values:

moolets

One or more Sigalisers/moolets which will be included in the merge group. Only Situations created by this Sigaliser or Sigalisers will be considered for merging with each other.

Type: String
Default: n/a

alert_threshold

The minimum number of Alerts that must be present in a cluster before it can become a Situation in the merge group.

Type: Integer
Default: 2

sig_similarity_limit

The measure of the similarity between two Situations before they are merged together. This value is the ratio of shared Alerts between two Situations to total unique Alerts in both Situations. For example, if two Situations share 50% of the same Alerts, the value would be 0.5.

Type: Integer
Default: 0.7

If you create a custom merge group for one or more Sigalisers, only Situations produced by the Sigalisers in the merge group will be considered for merging among themselves. Situations from Sigalisers outside of the defined merge group cannot be merged with any Situations in that group.

Field Behavior

When Cisco Crosswork Situation Manager merges two or more Situations, it updates the fields of the situations as follows:

Field	Old Situations	New Situation
Category	Superseded.	Created.

Created At	No change.	Time of merge.
Description	No change.	Merge of Situations [X, Y, Z] where X, Y, and Z represent the Situation IDs of the superseded Situations.
First Event Time	No change.	The First Event Time for the combined Situations.
ID	No change.	The next sequential Situation ID.
Last Change	No change.	The time that the merge took place.
Last Event Time	No change.	The value of the Situation in first position in the merge list.
Owned By	No change.	Default (none).
Participants	No change.	Default (none).
Process Impacted	No change.	Combined values.
Queue	No change.	The queue of the Situation in first position in the merge list.
Rating	No change.	Default (none).
Scope	No change.	Combined values.
Scope Trend	No change.	Combined values.
Services Impacted	No change.	Combined values.
Sev Trend	No change.	Combined values.
Severity	No change.	The highest severity of the merged Situations.
Status	Dormant.	Opened.
Story	Adopts ID of new Situation.	The Story ID is the same as the Situation ID of the new Situation.
Teams	No change.	All Teams monitoring the merged Situations.
Total Alerts	No change.	The sum of the Alerts of all merged Situations.
User Comments	No change.	Default (none).

Situation Room Plugins

- [Implementation](#)
- [Examples](#)
- [ServiceNow Integration](#)

You can add configurable third-party plugin tabs to the Situation Room in Cisco Crosswork Situation Manager that relate to the Situation. For example, you can link to a ServiceNow incident that is mapped to the Situation in question.

Cisco Crosswork Situation Manager requires a link_definition and custom_info column to enable the Situation Room Plugin.

The **Show ServiceNow Incident in the Situation Room** check box in System Administration, Integrations, ServiceNow adds the plugin automatically

Two use cases are:

- Conditional, based on contents of a field being present in the situation (if the field is left empty, the plugin will be disabled)
- Universal for all Situations

Further to these use cases, the link_definition can also have the same use cases:

- Conditional, taking attributes of the situation and passing to the linked application
- Generic, passing no details about the situation

Implementation

The **moogdb.sitroom_plugins** table has the following columns:

- the **title**
- its associated Situation field (from the **moogdb.sigs** table and can be **custom_info.<blah>**)
- the **link_definition** to use

```
mysql> describe moogdb.sitroom_plugins;
```

Field	Type	Null	Key	Default	Extra
title	varchar(32)	NO	PRI		
internal_name	varchar(255)	YES		NULL	
link_name	varchar(32)	YES	MU	NULL	L

3 rows in set (0.00 sec)

Examples

Assuming Situation 14 had already been linked with ServiceNow Incident **INC0000055** using the following SQL:

```
update moogdb.sigs set custom_info='{"servicenow_id":"INC0000055"}' where sig_id=14;
```

Enable a ServiceNow tab in the Situation Room

1. Define a **link_definition** to point to the ServiceNow URL, and use the \$valuedynamic placeholder to be replaced with the incident number when launched.

```
insert into moogdb.link_definitions (name, link) values ('servicenow', 'https://<your-server-here>/incident.do?sysparm_query=number%3D$value');
```

2. Add an entry into the **sitroom_plugins** table to define:

- the **tab**
- the Situation field to be used for **\$value**
- the **link_definition**

```
insert into moogdb.sitroom_plugins (title, internal_name, link_name) values ('ServiceNow', 'custom_info.servicenow_id', 'servicenow');
```

If desired, you can define multiple plugin tabs (with unique **title**) using the same or different Situation fields.

Addendum

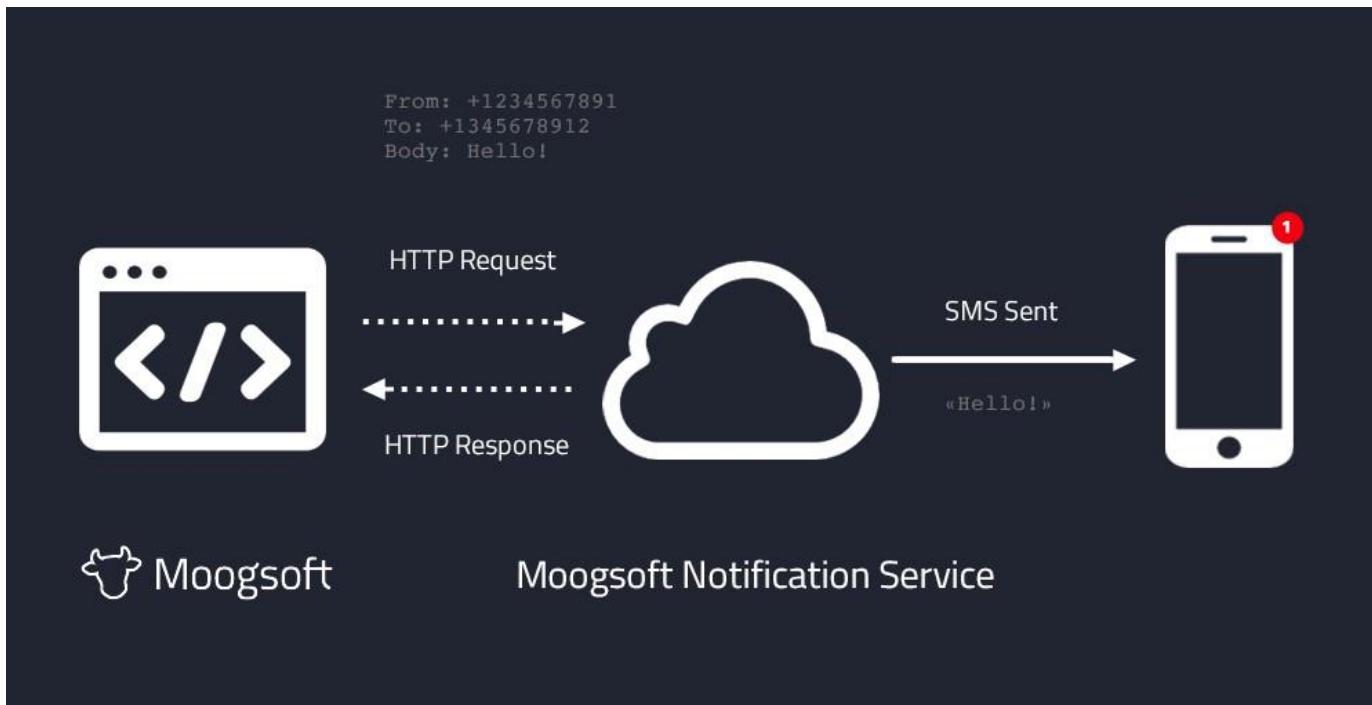
You must configure the browser to allow third party cookies from these URLs, otherwise, remote websites may not display within the tab area. To do this, either enable third-party cookies globally, or, allow the specific URLs to set third party cookies as exceptions.

ServiceNow Integration

Configure ServiceNow using the UI. See the Integrate ServiceNow section in [ServiceNow](#) for details.

SMS Configuration

Cisco Crosswork Situation Manager for Mobile uses a cloud-based notification service which allows messages to be sent via its REST API.



Any requests to the notification service's REST API will require authentication. For HTTP basic authentication, the username will be set to your **AccountSid** and the password is set to your **AuthToken**. See [Step 2](#) of Configuration below.

All text messages sent are limited to 1600 characters in length, any messages longer than this will be truncated.

Configuration

Please follow these steps to configure the mobile version to send and receive SMS messages:

1. Go to **System Settings > Users** and add phone numbers, including country codes, for all Cisco Crosswork Situation Manager users who you want to receive SMS messages.

Please note: All phone numbers must include the international country code in E.164 format. For example, +1, +44, +61 etc. See [International Dialing Codes](#)

← System Settings / Users

	PERSONAL	CONTACT	ROLES	COMPETENCIES	TEAMS
<input type="text"/>					
Administrator (admin)					
Moog (anon)					
Graze API User (graze)					
System Owner (super)					
super (superuser)					
user (user)					
Ali Admin (user1)					
Omar Operator (user2)					
Oscar Operator (user3)					
Olga Operator (user4)					
Olivia Operator (user5)					
Ingrid Implementer (user6)					

+ REVERT CHANGES SAVE CHANGES

Alternatively, to add your own number click on your user icon in the top right corner to open the **User Menu** then click **My Account...** and add a number next to 'Contact Number':

My Account ×

This basic information, such as your name, picture, email and contact number helps others to find you and makes it easier to get in touch.

A

FULL NAME:
This field is required

CHANGE **EMAIL:**
This field is required

REMOVE **CONTACT NUMBER:**

Normally you can leave these settings and use the system defaults. However, if you want to display a different timezone or date format then you can customize these as appropriate

TIME ZONE: ▾

DATE FORMAT: ▾

CANCEL **DONE**

2. Contact support@moogsoft.com or your Cisco Sales Engineer for your account details to enable SMS. These will include an account session ID, an authentication token and a phone number.

3. Open a terminal and open the Cisco Crosswork Situation Manager system.conf file (**\$CISCOHOME/config/system.conf**) and add your details in the following format (this can be added anywhere in the file):

```
"sms_sender":  
{  
    "account_sid" : "AB1234c6d91dfe4eef9b3899dkrw34aabbab2b", "auth_token" :  
    "4eee23d50de54333f6949366fe0882a4", "phone_number" : "+12345678910",  
    "character_limit" : 1600  
}
```

Please note: The snippet above is just an illustrative example and will not work if added to your system.conf file

4. Restart your Tomcat if it is still running. This will ensure all changes take effect.

5. By default, you will receive notifications about all invitations and assignments. This cannot be changed at present.

International Dialing Codes

The international dialing codes below are displayed in the correct format (+ followed by a number) required by our notification service:

Flag	Country	Code
	Australia	+61
	Brazil	+55
	China	+86
	Canada	+1
	France	+33
	Finland	+358
	Germany	+49
	Hong Kong	+852
	Hungary	+36
	India	+91
	Ireland	+353
	Israel	+972
	Italy	+39
	Japan	+81
	Luxembourg	+352

	Malaysia	+60
	Netherlands	+31
	Poland	+48
	Russia	+7
	Singapore	+65
	Slovakia	+421
	South Korea	+82
	Spain	+34
	Sweden	+46
	Switzerland	+41
	Taiwan	+886
	Turkey	+90
	U.K	+44
	U.S.A	+1

Troubleshooting

- Cisco Crosswork Situation Manager
- Installation and Upgrade Single-Host
- Installation for Non-Root Mobile Troubleshooting
- Cisco Crosswork Situation
- Manager Processes User
- Interface (UI) issues
- Processing Issues
- Error Messages & Status Codes

Cisco Crosswork Situation Manager Installation and Upgrade

YUM: Not installing correct version

If, when attempting to install Cisco Crosswork Situation Manager, an incorrect or outdated version is offered then your YUM cache may need cleaning. Try running:

yum clean all

and then attempt the install again.

YUM: HTTP Error 401 - Unauthorized

If an attempt to install Cisco Crosswork Situation Manager fails with an error such as:

```
https://<username>:<password>@speedy.moogsoft.com/repo/aiops/latest  
/repodata/repomd.xml: [Errno 14] HTTP Error 401 - Unauthorized
```

then check your <username> and <password> credentials are correct in the configured Moogsoft yum repository.

YUM: problem making ssl connection

If an attempt to install Cisco Crosswork Situation Manager fails with an error such as:

```
https://<username>:<password>@speedy.moogsoft.com/repo/aiops/latest  
/repodata/repomd.xml: [Errno 14] problem making ssl connection Trying other mirror.  
Error: Cannot retrieve repository metadata (repomd.xml) for repository: moogsoft-aiops. Please verify its  
path and try again
```

then you may need to update the **nss** packages on your server. Try running:

```
yum -y update nss
```

and then attempt the install again.

YUM: MySQL Conflict

If an attempt to install Cisco Crosswork Situation Manager fails with an error such as:

```
Running rpm_check_debug  
Running Transaction Test  
Transaction Check Error:  
  file /usr/lib64/mysql/libmysqlclient.so.16.0.0 from install of mysql-community-libs-compat-5.7.22-  
2.el6.x86_64 conflicts with file from package compat-mysql51-5.1.54-1.el6.remi.x86_64  
  file /usr/lib64/mysql/libmysqlclient_r.so.16.0.0 from install of mysql-community-libs-compat-5.7.22-  
2.el6.x86_64 conflicts with file from package compat-mysql51-5.1.54-1.el6.remi.x86_64  
Error Summary  
-----
```

then this is caused by a conflict with the MySQL libraries currently on the host.

Run the following bash commands to allow the product to be installed successfully:

```
echo "remove compat-mysql51" > /tmp/moog_yum_shell.txt  
echo "install mysql-community-libs-compat-5.7.22" >> /tmp/moog_yum_shell.txt  
echo "install mysql-community-client-5.7.22" >> /tmp/moog_yum_shell.txt
```

```
echo "install mysql-community-libs-5.7.22" >> /tmp/moog_yum_shell.txt
echo "install mysql-community-server-5.7.22" >> /tmp/moog_yum_shell.txt
echo "install mysql-community-common-5.7.22" >> /tmp/moog_yum_shell.txt
echo "groupinstall moogsoft" >> /tmp/moog_yum_shell.txt
echo "run" >> /tmp/moog_yum_shell.txt

cat /tmp/moog_yum_shell.txt | yum shell -y
```

The above error is most likely to occur on hosts which already have some MySQL components installed. i.e. the issue will most likely be seen when running the command `yum install moogsoft-dbon` a system with an existing MySQL install.

Nginx install issues

Error: Package: moogsoft-ui-6.0.0-123.x86_64 (moogsoft-aiops) Requires: nginx >= 1.14.0

If you encounter the following error when attempting to install Cisco Crosswork Situation Manager:

```
Requires: nginx >= 1.14.0
--> Package moogsoft-ui.x86_64 0:6.0.0-123 will be an update
--> Processing Dependency: nginx >= 1.14.0 for package: moogsoft-ui-6.0.0- 123.x86_64
--> Finished Dependency Resolution
Error: Package: moogsoft-ui-6.0.0-123.x86_64 (moogsoft-aiops) Requires: nginx >= 1.14.0
You could try using --skip-broken to work around the problem You could try running:
rpm -Va --nofiles --nodigest
```

then the nginx repo itself can be installed manually with the following command:

```
rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-
0.el7.ngx.noarch.rpm
```

The install can now take place.

Single-Host Installation for Non-Root

If you cannot access the UI from your host machine, check your firewall and if you're listening on the right ports:

1. To see if your firewall is enabled:

```
sestatus
```

If your firewall is disabled it returns:

SELinux status:	disabled
-----------------	----------

2. To disable an active firewall run:

```
setenforce 0
```

- To check whether a port is open you can run:

```
firewall-cmd --zone=public --query-port=8443/tcp firewall-cmd --zone=public --query-port=8080/tcp
```

- To open these ports run:

```
firewall-cmd --permanent --zone=public --add-port=8080/tcp firewall-cmd --permanent --zone=public --add-port=8443/tcp firewall-cmd --reload
```

Mobile Troubleshooting

Cisco Crosswork Situation Manager includes a self-signed certificate by default. If you want to use Cisco Crosswork Situation Manager for Mobile on an iPhone, you need to add a valid SSL certificate. This is because WebSockets do not work on iOS with self-signed certificates.

If a valid root CA certificate is not added, a 'Connection Error' appears at login and Cisco Crosswork Situation Manager for Mobile does not work.

Nginx SSL Configuration

To apply a valid certificate to Nginx, go to the `nginxconfig` folder and edit `moog-ssl.conf`:

```
cd common/config/nginx vi moog-ssl.conf vi moog-ssl.conf
```

Change the default self-signed certificate and key locations to point to the valid root certificate and key:

```
#ssl_certificate /etc/nginx/ssl/certificate.pem; #ssl_certificate_key  
/etc/nginx/ssl/certificate.key;  
  
ssl_certificate /etc/certificates/GeoTrust_Universal_CA.crt; ssl_certificate_key  
/etc/certificates/GeoTrust_Universal_CA.key;
```

Restart Nginx with this command:

```
systemctl restart nginx
```

Cisco Crosswork Situation Manager Processes

Required services for a functional production system

Service name	Description
apache-tomcat	Web server that contains the servlets that provide the Cisco Crosswork Situation Manager User interface
nginx	Web server that handles security, such as Cisco Crosswork Situation Manager login, php and https/ssl implementation

LAMs: socketlamd trapdlamd newreliclamd etc.	Link Access Modules used for data ingestion. Service names may differ At least one instance of a LAM is required for data feed. See Data ingestion
moogfarmd	Core Cisco Crosswork Situation Manager system application. See Event processing
mysqld	Database containing Cisco Crosswork Situation Manager data (moogdband moog_ref erencesschemas, etc.)
rabbitmq-server	Message system for Cisco Crosswork Situation Manager. See Core operation - MooMS
elasticsearch	Elasticsearch service for UI search feature.

To check the status, stop, start or restart a service run one of the following:

```
service <service-name> status
<service-name> stop
service <service-name>
service <service-name> start
service <service-name> restart
```

Location of installation and log files

See [Logging](#).

Generic Cisco Crosswork Situation Manager process not starting

No space left on the disk

- Check the file system with the following command:
`df -m`
- Look for partitions that are full

<Service> not found

Error while loading shared libraries

- Environmental variables may not be properly set up for your shell
- Run the environment and check the location set for `$MOOGSOFT_HOME`
- For information on setting the Cisco Crosswork Situation Manager shell environment, see [Installation and Upgrade guide](#)

moogfarmd not starting

Configuration parsing error

- +|No config present|+ message in `/var/log/moogsoft/moogfarmd.log`
- Points to a syntax error in `$MOOGSOFT_HOME/config/moog_farmd.conf`
- Check the config file for punctuation mistakes

Look for:

- Missing commas
- Unbalanced quotes
- Missing '{' or '}'

Use # to comment out code instead of /* and */

- Edit `moog_farmd.conf` and then restart the service

Rabbitmq related

Also see [Troubleshooting MooMS deployment](#).

Rabbitmq not starting - "No such user"

- No such user message in /var/log/rabbitmq/startup_err Check /etc/passwd
- for user rabbitmq with the following command: grep rabbitmq /etc/passwd
- If no user is found, add the following to /etc/passwd:
rabbitmq:x:491:488:RabbitMQ messaging server:/var/lib/rabbitmq/bin/bash

Rabbitmq not starting - "Failed to create aux thread"

- Failed to create aux thread message in /var/log/rabbitmq/startup_err
- This is most likely a **ulimit** issue for the **rabbitmq** user
- Check ulimits for the rabbitmq user by running (as root):

```
[root@ldev04 ~]# su - rabbitmq bash-4.1$  
ulimit -a  
core file size          (blocks, -c) 0  
data seg size           (kbytes, -d) unlimited  
scheduling priority     (-e) 0  
file size               (blocks, -f) unlimited pending  
signals                 (-i) 515675 max  
locked memory            (kbytes, -l) 64  
max memory size         (kbytes, -m) unlimited open files  
                           (-n) 1024  
pipe size                (512 bytes, -p) 8 POSIX  
message queues           (bytes, -q) 819200 real-  
time priority             (-r) 0 stack size  
                           (kbytes, -s) 10240  
cpu time                 (seconds, -t) unlimited max user  
processes                (-u) 1024 virtual  
memory                  (kbytes, -v) unlimited file locks
```

- The above example shows ulimits that are likely too low for RabbitMQ
- As per instructions [here](#) it may be appropriate to increase ulimits for "open files" and "max user processes" to at least **4096** for Development/QA environments and **65536** for Production environments

Unable to create RabbitMQ connection

- + Unable to create RabbitMQ connection+ message appearing in LAM, moogfarmd or apache-tomcat logs (see above for locations)
- This indicates that the process unable to connect to the MooMS zone in rabbitmq
- Check that the rabbitmq server is running with the following command:
service rabbitmq-server status
- If service is not running start it with the following command:
service rabbitmq-server start
- If the service is running, check that the zone used in Cisco Crosswork Situation Manager matches the vhost in rabbitmq List the zones (vhosts) added in rabbitmq with the following command:
rabbitmqctl list_vhosts
Check the MooMSsection in /usr/share/moogsoft/config/system.conf for the zone used in Cisco Crosswork Situation
- Manager If the zone is missing, add the zone (vhost) to rabbitmq manually (see [Troubleshooting MooMS deployment](#))
- Restart the process affected

LAMs fail to start from command line

If LAMs run from the command line or as a service result in the following error:

```
[root@moogbox2 bin]# ./socket_lam  
.socket_lam: error while loading shared libraries: libjvm.so: cannot open shared object file: No such file or  
directory
```

it may be because /usr/java/jdk1.8.0_171/jre/lib/amd64/server has not been added to the LD_LIBRARY_PATH.

- To run the LAMs via a command line, a change the LD_LIBRARY_PATH to be as follows (the default initfiles have this modification made):

```
export LD_LIBRARY_PATH=$MOOGSOFT_HOME/lib:/usr/GNUstep/Local/Library  
/Libraries:/usr/GNUstep/System/Library/Libraries:$JAVA_HOME/jre/lib/amd64  
/server
```

Generic LAM not starting

Configuration parsing error

- +|Unable to parse configuration file + message in /var/log/moogsoft/<lamd_name>.log
- This indicates a syntax error in the LAM configuration file
- Check the config file for syntax mistakes

Look for:
• Missing commas
• Unbalanced quotes

- Compare the config file to a default config file for the same LAM. Use the following command to locate differences in the files:
diff -y <current_lam>.conf <default_lam>.conf | less
- Edit <current_lam>.conf to resolve any syntax errors and restart the LAM

Connection refused error

- +failed to connect to [host:<port>]: Connection refused+ message in /var/log/moogsoft/<lamd_name>.log
- This means that the port specified in the LAM configuration file is already in use
- Use the following command to check that the LAM is not already started:
ps -ef | grep <lamd_name>
- Check that another process is not already bound to the port
- If required, edit the port setting for the LAM in the config file and restart the LAM

Unresolvable hostname error

- +|Host [hostname] unresolvable|+ message in /var/log/moogsoft/<lamd_name>.log
- The LAM is unable to resolve the host, or the hostname is incorrectly set
- In the LAM config file, check the address setting, and correct any errors
- Check that /etc/hosts has an entry for the hostname specified

Failed to open file error

- +|Failed to open file [<path_to_file>]|+ message in /var/log/moogsoft/<lamd_name>.log
- The LAM is unable to locate a file specified in the LAM configuration file
- Locate the missing file in \$MOOGSOFT_HOME/bots/lambots or \$MOOGSOFT_HOME/contrib
- Update the LAM config file with the correct file path, and restart the LAM

Generic LAM not processing

Syntax error in Presend filter

- In the LAM configuration file, locate the presend filter file name
- Either
With a javascript editor, check the code to locate any syntax errors,
Or, if installed, run:
node \$MOOGSOFT_HOME/bots/lambots/<path_to_filter_file>.js to locate the incorrect line in the code
Edit \$MOOGSOFT_HOME/bots/lambots/<path_to_filter_file>.js to resolve the error, and restart the LAM

Empty columns in alert lists

- Possibly indicates incorrect field mapping assignments
- Check field mappings at the bottom of the LAM configuration file

- Edit the config file to properly map the field to the column name, and then restart the LAM to test

Socket LAM not processing

Processing mode set incorrectly

- The LAM may be set to Servermode rather than Clientin the LAM configuration file. For an explanation of mode types, see Socket LAM
- Set the mode accordingly in the config file and restart the LAM

JSON feed not processing

- The JSON string is incorrectly formatted. For example, event data contains nested JSON
- Run the LAM in debugmode and look for nested JSON
- To resolve, either modify the event data, or edit the presend filter to match values in the nested JSON

Logfile LAM not starting

No input file

- +|Could not stat file [-1] error: [Bad file descriptor]|+ message in /var/log/moogsoft/<lamd_name>.log
- The Logfile LAM cannot locate the log file to read
- In the LAM configuration file, check the targetsetting and confirm the file path to the target log file

REST LAM not starting

Missing SSL path

- +|No file path specified|+ message in /var/log/moogsoft/<lamd_name>.log
- In the LAM configuration file, check the use_ssl setting
- If using SSL, (use_ssl set to true), then check the following are properly set:
 - path_to_ssl_files
 - ssl_key_filename
 - ssl_cert_filename
- If not using SSL, set use_ssl to false

Nginx fails on startup if IPv6 not configured

Comment out the following references in two conf files:

1. go to /etc/nginx/conf.d
2. edit out IPv6 references with a #:

conf file	section
moog-default.conf	listen 80 default_server; #listen [::]:80 default_server;
moog-ssl.conf	listen 443 ssl default_server; # listen [::]:443 ssl;

User Interface (UI) issues

Unavailable UI Login Page

- Check that port 443 is not being blocked by the firewall on the server.
- Check that the **nginx** service is running with command:

```
[root@moogbox2 ~]# service nginx status nginx (pid
42356) is running
```

- Check that nginx is listening on port 443 - example expected output:

```
[root@moogbox2 ~]# netstat -anp|grep 443
tcp        0      0  0.0.0.0:443                           0.0.0.0:
*          LISTEN     42356/nginx
tcp        0      0  :::443                               :::
*          LISTEN     42356/nginx
```

Login fails with "You could not be logged in. Please try again."

Apache-tomcat service not running

- Check the apache-tomcat service is running with the following command:
service apache-tomcat status

Communication problem between the UI and MySQL database

- Check the MySQL service is running with the following command:
service mysqld status
- If MySQL is running on a different server, is it reachable from the Moog Cisco Crosswork Situation Manager web server and have the required grants been applied?

Authentication problem between the UI and MySQL database

- Does your user exist in the system?
Check in the MySQL moogdb.users table Is
- the username and password correct?

Search/Elasticsearch Troubleshooting

See [Search and Indexing](#).

ElasticSearch not running or generating errors (such as MySQL connection problems)

- Check that the elasticsearch service is running:

```
[root@moogbox2 ~]# service elasticsearch status elasticsearch (pid
39596) is running
```

- Any errors will be written to /var/log/elasticsearch/elasticsearch.log

Tomcat cannot connect to ElasticSearch

- Check /usr/share/apache-tomcat/logs/catalina.out for any errors when attempting a search from the UI

Cron job errors

- Check that cron job that runs the **moog_indexer** (created by the moog_init_search.sh script to re-index against the Cisco Crosswork Situation Manager database on a once-a-minute basis) exists and is not generating any warnings or errors
- Running crontab -l will list the configured cron jobs
- Errors are in /var/log/cron
- Depending on the interval at which Elastic re-indexes against the Cisco Crosswork Situation Manager database, it is possible that new Alerts, Situations, Thread or Comments have not yet been indexed, and so will not be searchable
- To change the interval manually, run the following command:
crontab -ed

Elasticsearch fails to start with /tmp directory permission problems

Elasticsearch fails to start with "java.lang. UnsatisfiedLinkError: /tmp/jna--<blah>" error. For example:


```
[2017-08-07T14:14:31,173][WARN ][o.e.b.Natives] unable to load JNA
native support library, native methods will be disabled. java.lang.UnsatisfiedLinkError: /tmp/jna--1985354563
/jna3872404023206022895.tmp: /tmp/jna--1985354563/jna3872404023206022895.
tmp: failed to map segment from shared object: Operation not permitted at
java.lang.ClassLoader$NativeLibrary.load(Native Method) ~[?:1.8.0_171]
at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1941) ~[?:1.8.0_171]
at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1824) ~[?:1.8.0_171]
at java.lang.Runtime.load0(Runtime.java:809) ~[?:1.8.0_171] at
java.lang.System.load(System.java:1086) ~[?:1.8.0_171]
at com.sun.jna.Native.loadNativeDispatchLibraryFromClasspath (Native.java:851) ~[jna-
4.2.2.jar:4.2.2 (b0)]
at com.sun.jna.Native.loadNativeDispatchLibrary(Native.java:826) ~ [jna-4.2.2.jar:4.2.2 (b0)]
at com.sun.jna.Native.<clinit>(Native.java:140) ~[jna-4.2.2.jar:
4.2.2 (b0)]
at java.lang.Class.forName0(Native Method) ~[?:1.8.0_171] at
java.lang.Class.forName(Class.java:264) ~[?:1.8.0_171]
at org.elasticsearch.bootstrap.Natives.<clinit>(Natives.java:45) [elasticsearch-5.6.9.jar:5.6.9]
at org.elasticsearch.bootstrap.Bootstrap.initializeNatives (Bootstrap.java:104)
[elasticsearch-5.6.9.jar:5.6.9]
at org.elasticsearch.bootstrap.Bootstrap.setup(Bootstrap.java:203) [elasticsearch-5.6.9.jar:5.6.9]
at org.elasticsearch.bootstrap.Bootstrap.init(Bootstrap.java:333) [elasticsearch-5.6.9.jar:5.6.9]
at org.elasticsearch.bootstrap.Elasticsearch.init(Elasticsearch.java:121) [elasticsearch-
5.6.9.jar:5.6.9]
at org.elasticsearch.bootstrap.Elasticsearch.execute(Elasticsearch.java:112) [elasticsearch-
5.6.9.jar:5.6.9]
at org.elasticsearch.cli.SettingsCommand.execute(SettingsCommand.java:54) [elasticsearch-
5.6.9.jar:5.6.9]
at org.elasticsearch.cli.Command.mainWithoutErrorHandler(Command.java:122) [elasticsearch-
5.6.9.jar:5.6.9]
at org.elasticsearch.cli.Command.main(Command.java:88) [elasticsearch-
5.6.9.jar:5.6.9]
at org.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:89) [elasticsearch-
5.6.9.jar:5.6.9]
at org.elasticsearch.bootstrap.Elasticsearch.main(Elasticsearch.java:82) [elasticsearch-
5.6.9.jar:5.6.9]
```

This is most likely due to the /tmp mount having the noexec directive.

Solution:

Either:

- Remove the noexec directive from the /tmp mount (if practical to do so)

```
sudo mount /tmp -o remount,exec
```

or:

b) In file: /etc/sysconfig/elasticsearch set:

```
ES_JAVA_OPTS="-Djna.tmpdir=/var/lib/elasticsearch/tmp"
```

Restart the elasticsearch service after either of the above changes.

Processing Issues

No Alerts created

License not applied

- Ensure product license has been applied.

Rabbitmq not running

- Check that the rabbitmq server is running with the following command:
service rabbitmq-server status

AlertBuilder not started

- Check that the run_on_startup setting for AlertBuilder in \$MOOGSOFT_HOME/config/moog_farmd.conf is set to true
Check
• /var/log/moogsoft/moogfarmd.log

AlertBuilder misconfiguration

- Check the AlertBuilder Moollet for syntax errors or errors in logic

LAM misconfiguration

- Check that the LAM is correctly parsing/mapping the data feed
- Check that the LAM is not doing any post-event processing that may be filtering out the events out in the associated LAMBot

No Situations Created

License not applied

- Ensure product license has been applied

Sigaliser Moollet not running

- In \$MOOGSOFT_HOME/config/moog_farmd.conf check that the run_on_startup setting for the sigaliser used is set to true

Incorrectly set process_output_of

- In \$MOOGSOFT_HOME/config/moog_farmd.conf check that the process_output_of setting for the sigaliser used is listing the correct Moollet (for example, AlertBuilder)
- Also check that the Moollet listed under process_output_of is running

Sigaliser Moollet configuration is too restrictive or too open

If Moollet settings are too restrictive or too open they may not produce Situations. See Sigaliser Moollet for more information

Error Messages & Status Codes

HTTP Status Codes

The following list describes the HTTP status codes used by Cisco Crosswork Situation Manager:

Code	Description	Application Usage
200	Success	This status will be sent to the browser when a request has been processed 'successfully'.
202	Accepted	This status will be sent to the browser when the request was well-constructed but the server was unable to process it.
400	Bad Request	This status will be sent to the browser when the request parameters are invalid or missing.
401	Unauthorized	This status code returned when the user is not authenticated.
403	Forbidden	This status code returned when the user is forbidden from performing a specific action.
404	Not Found	This status code will be sent to browser when a requested servlet path could not be found.
409	Conflict	Some conflict has occurred that can potentially be resolved by the user e.g. change a template name
500	Internal Server Error	This status will be sent to the browser when there is some unexpected exception or problem on the server side.
503	Service Unavailable	Used when the server is overloaded (currently used by tool runner when no execution threads are available)

Application Status Codes

The application status codes present in the response payload are defined in the following ranges:

Please note: One range may map to more than one HTTP status

Code Range	Description	HTTP Status
1000-1999	System Errors	500, 503
2000-2999	Validation Errors	400
3000-3999	Security Errors	401, 403
4000-4999	Conflict Errors	409

System Error Status Codes

The following system error status codes are currently defined:

Code	Description	HTTP Status
1000	General server error	500
1001	Service Unavailable	503

1002	DB Unavailable	500
1003	Service Busy	202

Validation Error Status Codes

The following validation error status codes are currently defined:

Code	Description	HTTP Status
2000	General validation error. The actual parameter specific error will be defined in the additional body.	400
2001	Parameter missing (not included in post/get) or no value provided (parameter included but empty value)	400
2002	Parameter format error (a value that could not be converted to desired type)	400
2003	Illegal value (a value that is not in the list of allowed values)	400
2004	Parameter out of range (a value that is outside the range of allowed values)	400

Security Error Status Codes

The following security error status codes are currently defined:

Code	Description	HTTP Status
3000	General security error	401
3001	User not authenticated	401
3002	User not permitted to perform action	403
3003	Header auth token not authenticated	401

Conflict Error Status Codes

The following conflict error status codes are currently defined:

Code	Description	HTTP Status
4000	General conflict error	409
4001	Duplicate name e.g. Duplicate template name, Invite already sent	409

Content Error Status Codes

The following content error status codes are currently defined:

Code	Description	HTTP Status
5000	General content error	500

5001	Content not supported	500
5002	Error decoding content	500
5003	Content accepted but cached, processing not guaranteed	202
5004	Server being ahead of the UI (in terms of Moog Version)	404
5005	UI being ahead of the UI (in terms of Moog Version)	40

JSON Response Body (Non-HTTP 200)

The response body will contain the detailed application error status information. The format of the JSON is as follows:

Please note: The additional information is optional for some status codes

```
{ "message" : "User friendly message", "statusCode" :  
<appStatusCode>, "additional" : <jsonAdditionalInfo> }  
-or-  
{ "message" : "User friendly message", "statusCode" :  
<appStatusCode> }
```

The high-level user friendly message will take the following format:

"<operation> failed due to <something>"

e.g. "Create situation failed due to invalid parameters"

JSON Addition Information

The format of the JSON additional information will depend on the the HTTP status code and application status code. For certain status codes the additional information may not be present. Although the HTTP status code (400 or 500) is enough to determine the additional information structure it is probably best to use the application status code (as in the future more than one range could apply to a single HTTP status code).

- HTTP 400 -> Application Status Code 2000-2999
- HTTP 409 -> Application Status Code 4000-4999
- HTTP 500, 503 -> Application Status Code 1000-1999

Application Status Code [1000-1999]

Optional additional information

A debug message typically containing the exception message:

```
{ "debugMessage" : "java.net.ConnectException: Connection refused" }
```

Application Status Code [2000-2999] or [4000-4999]

Mandatory additional information

List of the bad parameter(s) and the error information:

```
[ { "name" : "parameterName", "value" :  
<badValue>,  
"errorCode" : <errorCode> },
```

```
{...},  
{...},  
...]
```

Nginx Error Messages

97: Address family not supported by protocol

This is an IPv6 error. Either configure IPv6 or comment it out.

Comment out the following references in two conf files:

- Go to /etc/nginx.conf.d
- Edit out IPv6 references with a hash #:

Conf file	Section
moog-default.conf	listen 80 default_server; #listen [::]:80 default_server;
moog-ssl.conf	listen 443 ssl default_server; # listen [::]:443 ssl;

502 Bad Gateway (nginx/1.14.0)

Situation: Fails to load resource and server responds with a status of 502 (Bad Gateway).

Resolution: Check that you have used the correct components for the version you are installing.

Please note: The SERVER 2 components changed slightly between Cisco Crosswork Situation Manager versions 6.0.0 and 6.1.0

```
moog_init_mooms.sh -z MY_ZONE moog_init_lams.sh -bz  
MY_ZONE -d SERVER1:3306 moog_init_search.sh -sd  
SERVER1:3306  
moog_init_server.sh -bz MY_ZONE -d SERVER1:3306  
moog_init_ui.sh -otwxfz MY_ZONE -d SERVER1:3306
```

URL-based Filters

Introduction

You can create URLs to open filtered Alerts and Situation Views. This is useful for context linking from a third-party application directly to Cisco Crosswork Situation Manager.

For example, context linking a device in a topology mapping product to an Alert View for that device. It also allows creation of powerful dynamic Alert and Situation Client Tools.

Creating a URL (Basic)

To create a valid URL, it must contain the location of Cisco Crosswork Situation Manager, the type of View and the basic filter query syntax to define the filter. When creating a new URL, it should contain the following components:

Component	Example	Description

The host server	https://<servername>/	The protocol and name of the host server
The view type	[#alerts #situations #situationAlerts]	Defines whether an Alert View, Situation View or Alerts assigned to Situations are displayed
The filter type	?[filterededitor=basic filterededitor=advanced]	Defines whether the filter will use basic or advanced query syntax Please note: Place a question mark (?) at the start of your query parameter and separate each subsequent parameter with an ampersand (&)
The parameters	&[filter-active_sig_list= filter-alert_id= filter-agent=]	These are the parameters, operators and values used to define in the filter. E.g. &filter-alert_id=12, &filter-count=5 etc. For all available parameters click here Please note: All parameter names must be prefixed by 'filter-'

Basic Example

The example below shows a URL-based filter using basic filter syntax:

```
https://<servername>/#situationAlerts/172?filterededitor=basic&filter-type=CPUHigh&filter-severity=2
```

A breakdown of this URL's components are described in the table below:

Example Component	Description
https://<servername>/	The host server for Cisco Crosswork Situation Manager
#situationAlerts/172	Specifies an Alert View of all Alerts assigned to Situation 172
?filterededitor=basic	Specifies that you want to use basic filter query string
&filter-type=CPUHigh&filter-severity=2	Defines the filter will display all Alerts with the Alert Type 'CPUHigh' and that have the severity of 'Warning' (severity level 2)

Severity ↓	Type	First Event Time	Last Event Time	Count	Description
Warning	CPUHigh	22:01:09 20/04/20...	09:52:59 24/04/20...	99	CPU Exceeds 90%
Warning	CPUHigh	22:01:09 20/04/20...	09:53:00 24/04/20...	104	CPU Exceeds 90%
Warning	CPUHigh	22:01:09 20/04/20...	09:53:00 24/04/20...	106	CPU Exceeds 90%
Warning	CPUHigh	22:01:08 20/04/20...	09:52:52 24/04/20...	100	CPU Exceeds 90%
Warning	CPUHigh	22:01:08 20/04/20...	09:52:56 24/04/20...	105	CPU Exceeds 90%

Please note: When using the URLs, if not currently logged into Cisco Crosswork Situation Manager, users are prompted to login. Alerts and Situation Views opened are active and are updated with the latest data, with filter and action and navigation functions available as normal for that user

Custom_info Field

You can filter URL-based filters for custom_info fields if you have added any to your instance of Cisco Crosswork Situation Manager.

Please note: For more information on adding custom_info fields see [Custom Info](#)

The example below shows a URL-based filter

```
https://<servername>/#situations?filtereditor=basic&filter-custom_info.something_new=5
```

Component	Description
https://<servername>/	The host server for Cisco Crosswork Situation Manager
#situations?	Specifies an Alert View of all Alerts assigned to Situation 172
?filtereditor=basic	Specifies that you want to use basic filter query string
&filter-custom_info.something_new=5	Defines that you filter the custom_info field 'something_new', this must be column field. The basic filter treats the field as text and uses 'matches' rather than 'equals'

Creating a URL (Advanced)

To create a valid URL using the Advanced Filter, it must contain the same components as before but they must be URL-encoded. See "Advanced Filter Syntax" on [Filter Search Data](#)

***Please note:** To encode a filter query, open DevTools (right-click and select **Inspect**) and go to **Console**:

Type 'encodeURI', insert the query in brackets and then double quotation marks:

Console Entry

```
encodeURI ("Severity = 'Warning' AND Type = 'DBFail'")
```

Press **Enter** to continue and encode the entry:

Encoded Result

```
"Severity%20=%20'Warning'%20AND%20Type%20=%20'DBFail'"
```

Advanced Example

The example below shows a URL-based filter using advanced filter query syntax:

```
https://<servername>/#situationAlerts/172?filtereditor=advanced&filter-query=Severity%20=%20'Critical'%20AND%20Severity%20=%20'Minor'
```

This URL can be broken down into the following components:

Component	Description
https://<servername>/	The local host or location of Cisco Crosswork Situation Manager
#situationAlerts/172	This specifies an Alert View of all Alerts assigned to Situation 172
?filterededitor=advanced	This specifies that you want to use advanced filter query syntax
&filter-query=Severity%20=%20'Critical'%20AND%20Severity%20=%20'Minor'	This defines the filter will display all Alerts with 'Critical' and 'Minor' severity

	SEVERITY ↓	TPS LEVEL	HOST	TYPE	OWNED BY	FIRST EVENT TIME
<input type="checkbox"/>	✖ Critical		ar0-acmevoip.enbrs.isp.acme.co...	TCP Connection S...		22:20:59 20/04/20...
<input type="checkbox"/>	✖ Critical		qfn-onx-lbb-1.acme.com (ssh)	Generic Link Status		22:18:22 20/04/20...
<input type="checkbox"/>	✖ Critical		vm0.negf.isp.acme.com (ssh)	isamv-loss-over-ca...		22:18:18 20/04/20...
<input type="checkbox"/>	⚠ Minor		vm7.lsput.isp.acme.com (ssh)	isamv-lcard-unrea...		22:20:37 20/04/20...
<input type="checkbox"/>	⚠ Minor		ge0.fc0.nme.blon.isp.acme.com	Anomalyflag		22:07:40 20/04/20...

Custom_info field

You can also filter custom_info fields if you have added any to your instance of Cisco Crosswork

Situation Manager. Example:

```
https://<servername>/#situations/?filterededitor=advanced&filter-query=%
60Something%20new%60%20MATCHES%20%225%22
```

Example Component	Description
https://<servername>/	The host server for Cisco Crosswork Situation Manager
#situations?	This specifies an Alert View of all Alerts assigned to Situation 172
?filterededitor=advanced	This specifies that you want to use advanced filter query string
&filter-query=%60Something%20new%60%20MATCHES%20%225%22	This defines that you filter the custom_info field 'something_new'. This must be a column field.

Popout Shortcut

There is a method to quickly popout a URL for Situation Alerts.

Please note: The **Popout** action can only be performed on Alerts that are assigned to Situations at present

To do this, go to the **Situation Room** for the Situation you are interested in and select the **Alerts** tab.

Create a Basic or Advanced filter then go to **Tools > Popout**:

Severity ↓	TPS Level	Host	Type	Owned By	First Event Time	Last Event Time
<input type="checkbox"/> Major		auto_server 5 (ssh)	DBTrans		22:01:24 20/04/20...	09:52:44 24/04/20...
<input type="checkbox"/> Major		auto_server 6 (ssh)	DBTrans		22:01:01 20/04/20...	09:52:39 24/04/20...

This will launch a new browser tab with the URL for the filter. See example below:

```
https://<servername>/#situationAlerts/172?filter-severity=4&filter-type=DBTrans&filterededitor=basic&sort=severity%3ADESC%2Calert_id%3ADESC
```

The filter URL will include the default sort order of Alerts in descending severity order then by descending Alert ID:

sort=severity%3ADESC%2Calert_id%3ADESC

Please note: If using the Popout method to generate a URL-based filter with advanced filter query syntax it will be automatically URL encoded

Use in Alert and Situation Client Tools

You can create powerful dynamic Alert and Situation Client Tools. URLs created using this mechanism can be added to Alert and Situation client tools, so that their functionality is available from right-click menus in Situation and Alert Views in Cisco Crosswork Situation Manager.

Examples

In an Alert client tool, with the **HTTP Method** GET selected, the following code in the **URL** field shows an Alert View with all Alerts that have the same host as the Alert the tool was run from:

```
https://<servername>/#alerts?filterededitor=basic&filter-source=$source
```

In a Situation client tool, with the **HTTP Method** GET selected, the following code in the **URL** field shows a Situation View of all Situations that are impacting the same services at the Situation the tool was run from:

```
https://<servername>/#situations?filterededitor=basic&filter-service_list=$service_list
```

Parameters

The tables below list the available parameters and the associated operators for Alerts and Situations:

Please note: The operators listed below are to be used in the filter query only, prior to using either the Popout or URI-encoding to form your URL

Alert Parameters

--	--	--

UI/Display Name	Filter Parameter	Operator
Active Situations	active_sig_list	IN
Alert Id	alert_id	> >= < <= != =
Agent Name	agent	MATCHES
Agent Host	agent_location	MATCHES
Class	class	MATCHES
Count	count	> >= < <= != =
Description	description	MATCHES
Entropy	entropy	> >= < <= != =
External ID	external_id	MATCHES
First Event Time	first_event_time	>= AND <=*
Host	source	MATCHES
Internal Last Event Time	int_last_event_time	>= AND <=*
Last Change	last_state_change	>= AND <=*
Last Event Time	last_event_time	>= AND <=*
Manager	manager	MATCHES
Owned By	owner	IN
Severity	severity	IN
Significance	significance	IN
Situations	sig_list	IN
Source ID	source_id	MATCHES
Status	state	IN
Type	type	MATCHES

Situation Parameters

UI/Display Name	Filter Parameter	Operator
Category	category	MATCHES
Created At	created_at	>= AND <=*
Description	description	MATCHES
First Event Time	first_event_time	>= AND <=*
ID	sig_id	> >= < <= != =
Last Change	last_state_change	>= AND <=*
Last Event Time	last_event_time	>= AND <=*
Owned By	owner	IN
Participants	participants	> >= < <= != =
Process Impacted	process_list	CONTAINS
Scope Trend	delta_entities	>0 <=0
Services Impacted	service_list	CONTAINS
Sev Trend	delta_priority	>0 <=0
Severity	severity	IN
Status	state	IN
Story	story_id	> >= < <= != =
Teams	teams	IN
Total Alerts	total_alerts	> >= < <= != =

User Comments	user_comments	> ≥ < ≤
---------------	---------------	------------------

!=

=

***Please note:** These parameters have operators defining two times, 'from' and 'to' separated by a colon. These times need to be epoch/Unix times:

E.g. First Event Time: From May 3, 2017 00:45:00 to May 3, 2017 00:45:00 would appear as ('First Event Time` >= 1493768700) AND ('First Event Time` <= 1493768700) in advanced filter query syntax and in the URL, this will appear as follows:

?filter-first_event_time=1493768700000%3A1493768700000

Valid SSL Certificates

Cisco Crosswork Situation Manager includes a self-signed certificate by default. If you want to add your own certificates to Nginx, follow the instructions below.

A valid SSL certificate is required if you want to use Cisco Crosswork Situation Manager for Mobile on an iPhone. This is because WebSockets do not work on iOS with self-signed certificates. If a valid root CA certificate is not added, a 'Connection Error' appears at login and Cisco Crosswork Situation Manager for Mobile does not work.

For more information, see the [Nginx documentation](#).

Add a Valid Certificate

To apply a valid certificate to Nginx, go to the nginxconfig folder and edit moog-ssl.conf:

```
cd common/config/nginx vi  
moog-ssl.conf
```

Change the default self-signed certificate and key locations to point to the valid root certificate and key:

```
#ssl_certificate /etc/nginx/ssl/certificate.pem; #ssl_certificate_key  
/etc/nginx/ssl/certificate.key;  
  
ssl_certificate /etc/certificates/your_company_certificate.crt; ssl_certificate_key  
/etc/certificates/your_company_certificate.key;
```

Reload Nginx with the command:

```
systemctl reload nginx
```