



Cisco Crosswork Situation Manager 7.2.x Implementor Guide

Powered by Moogsoft AIOps 7.2



Table of Contents

Implementer Guide	4
Cisco Crosswork Situation Manager 7.2.0 Supported Environments	5
Operating Systems	5
Browsers	5
Supported Third-Party Software	6
Integration Support.....	6
Sizing Recommendations.....	8
Small	9
Medium.....	9
Large	9
Install and Upgrade Cisco Crosswork Situation Manager.....	11
Cisco Crosswork Situation Manager Packages	11
Deployment Options	12
AWS Marketplace Installation.....	12
Post Install Validation.....	14
Encrypt Database Communications.....	14
Change Passwords for Default Users.....	16
Scale Your Cisco Crosswork Situation Manager Implementation	19
Uninstall Cisco Crosswork Situation Manager.....	20
Control Cisco Crosswork Situation Manager Processes	24
RPM Installation and Upgrade	27
Pre-Install Cisco Crosswork Situation Manager - Offline RPMs	27
Install Cisco Crosswork Situation Manager on a Single Host with RPM	31
Distributed Installation - RPM	32
Pre-install Cisco Crosswork Situation Manager	36
Upgrade - RPM Deployments.....	39
High Availability Overview.....	62
High Availability Configuration Hierarchy.....	65
High Availability Reference Architecture	66
High Availability for Third Party Component Dependencies	68
Single Site High Availability	69
HA Control Command Reference	74
Tarball Installation and Upgrade	74
Distributed Tarball Installation	74



Single Host Tarball Installation	78
Upgrade - Tarball Deployments	82
Post-installation Setup	104
Apply Valid SSL Certificates	104
Moog Encryptor	105
Configure External Authentication.....	107
Configure the Message Bus.....	130
Configure Search and Indexing.....	143
Configure Logging	147
Configure Services to Restart	158
Configure the Tool Runner	162
Configure SMS Notifications.....	169
Enable Situation Room Plugins.....	170
Import a Topology.....	172
Configure Historic Data Retention	174
Archive Situations and Alerts.....	181
How Archiving Works.....	182
Launch the Archiver	182
Archive Loose Alerts	182
Archive Filtered Situations and Alerts.....	183
Delete Situations, Alerts and Statistical Data	183
Archive File Names and Structure	183
Usage Tips	184
Archiver Command Reference	184
Probable Root Cause	186
How does PRC work?	187
How does Cisco Crosswork Situation Manager learn?	187
PRC Column	187
URL-based Filters.....	187
Creating a URL (Basic).....	187
Creating a URL (Advanced)	190
Popout Shortcut	191
Use in alert and Situation Client Tools.....	192
Parameters	193
System Configuration	196



Configure your System.....	197
System Configuration Reference.....	204
Configure Data Ingestion.....	216
Custom Info.....	217
Data Parsing.....	219
Severity Reference	226
Configure Data Processing.....	227
Services.....	228
Alert Processing	228
Enrichment	304
Moogfarmd Reference	309
Services.....	326
Situation Merge Behavior.....	330
Monitor and Troubleshoot Cisco Crosswork Situation Manager.....	333
Processing Metrics	333
Graze getSystemStatus Endpoint.....	335
Moogfarmd Health Logging.....	335
Tomcat Servlet Logging	336
Database Pool Diagnostics.....	337
MySQL Slow Query Logging	339
RabbitMQ Admin UI.....	341
Other Utilities	341
Troubleshooting Performance Problems	343
Cisco Crosswork Situation ManagerComponent Performance.....	348
Troubleshooting.....	353

Implementer Guide

The Implementer Guide contains instructions to help you install and configure Cisco Crosswork Situation Manager.

To install the system and handle common post-installation setup, see [Install and Upgrade Cisco Crosswork Situation Manager](#) and [Post-installation Setup](#).

After you have the base system up and running, you can begin to ingest event data from your monitoring sources. [Integrations](#) covers most integrations topics. You can find



some detail on some common configuration tasks for data ingestion under [Configure Data Ingestion](#).Integrations

Much of the value of Cisco Crosswork Situation Manager comes from its ability to process your raw event data, deduplicate the events, and transform the data into alerts that comprise Situations. It is critical to configure the system to create meaningful Situations for you and to present the Situations to the right teams. Figuring out your needs for your Situation design will help you make decisions about the right data processing choices for you.

Based upon your Situation design choices and the type of data available from your monitoring sources, you can follow the [Clustering Algorithm Guide](#) to choose the correct clustering algorithms for your system. Then you have several options to [Configure Data Processing](#) to achieve your goals. See also the [Administrator Guide](#).Clustering Algorithm GuideAdministrator Guide

To keep your system running and healthy, see [Monitor and Troubleshoot Moogsoft AIOps](#).

Cisco Crosswork Situation Manager 7.2.0 Supported Environments


The following operation systems, browsers and third-party software are either supported or are required in order to run Cisco Crosswork Situation Manager.

Any operating systems and browsers not listed in the sections below are not officially recommended or supported.

Operating Systems

You can run Cisco Crosswork Situation Manager on the following versions of [Red Hat Enterprise Linux®](#)(RHEL) and [CentOS Linux](#):

OS	Versions
----	----------

 CentOS	v7
--	----

 RHEL	v7
--	----

Note

No other Linux distributions are currently supported

Browsers

You can use the following browsers for the Cisco Crosswork Situation Manager UI:

Version

 Chrome	Latest
--	--------



 Firefox Latest

 Safari Latest

 Edge Latest

 IE v11

Supported Third-Party Software

The latest default installation of Cisco Crosswork Situation Manager comes with the following third-party applications:

Application	Version
Apache Tomcat®	v9.0.13
Elasticsearch	v5.6.9
MySQL Community Server	v5.7.22
Nginx	v1.14.0 or above
RabbitMQ	v3.7.4

Other supported application packages include:

Application	Version
Erlang	v20.1.7
JDK	OpenJDK 11.0.2
Apache Tomcat® Native	v1.2.21 or above

Integration Support

The following table outlines the vendor supported integrations for the current version of Cisco Crosswork Situation Manager alongside the corresponding supported software versions.

Integrations support IPv6 connectivity.

Integration Version	Supported Software / Version
Ansible Tower Integration v1.10	Ansible Tower v3.0, 3.1
Apache Kafka Integration v1.11	Apache Kafka v0.9, 1.1
AppDynamics Integration v2.2	AppDynamics v4.0, 4.1



AWS CloudWatch Integration v2.0	aws-java-sdk v1.11
AWS SNS Integration v1.2	AWS SNS v2016-06-28
CA UIM Integration v1.8	CA Nimsoft UIM v8.4
CA Spectrum Integration v2.1	CA Spectrum v10.2
Cherwell Service Management Integration v1.5	Cherwell v9.3
Datadog Polling Integration v1.2	Datadog v2018
Datadog Webhook Integration v1.11	Datadog v5.21
Dynatrace APM Plugin Integration v1.8	Dynatrace v6.5, 7.0
Dynatrace APM Polling Integration v2.2	Dynatrace v6.5, 7.0
Dynatrace Notification Integration v1.5	Dynatrace v6.5
Dynatrace Synthetic Integration v1.12	Dynatrace Synthetic v2017
Email Integration v2.5	IMAP, IMAPS, POP3, POP3S
EMC Smarts Integration v1.3	EMC Smarts v9.5
ExtraHop Integration v1.2	ExtraHop v7.4, 7.5
FluentD Integration v1.10	FluentD v0.12
Grafana Integration v1.2	Grafana v4.6 or above
HP NNMi Integration v2.5	HP NNMi v10.2
HP OMi Plugin Integration v1.7	HP OMi v10.1
HP OMi Polling Integration v2.5	HP OMi v10.1
JIRA Service Desk Integration v1.10	JIRA Service Desk v7.6
JIRA Software Integration v1.10	JIRA Software v7
JMS Integration v1.11	ActiveMQ v5.14, JBoss v10, WebLogic v12.0
Microsoft Azure Integration v1.2	Microsoft Azure Monitor v2018
Microsoft Azure Classic Integration v1.2	Microsoft Azure Classic v2018
Microsoft SCOM Integration v2.5	Microsoft SCOM v2012, 2016
Microsoft Teams Integration v1.0	Microsoft Teams v1.2.00.3961
Nagios Integration v2.10	Nagios vXI



New Relic Integration v1.10	New Relic v2016
New Relic Polling Integration v2.0	New Relic v2.3
New Relic Insights Polling Integration v1.0	New Relic v2.3
Node.js Integration v1.9	Node.js v1.6
NodeRED Integration v1.9	Nagios Red v016, 017
OEM Integration v2.3	Oracle Enterprise Manager v12c, 13c
Pingdom Integration v1.9	Pingdom v2017
Remedy Integration v1.7	Remedy v9.1
ServiceNow Integration v4.3	ServiceNow vHelsinki, Istanbul, Jakarta, Kingston
SevOne Integration v1.4	SevOne v5.4
Slack Integration v1.7	Slack v3.1
SolarWinds Integration v3.2	SolarWinds v11.5, 12.2
Splunk Integration v2.5	Splunk v6.5, 6.6, 7.0
Sumo Logic Integration v1.1	Sumo Logic v2018
VMware vCenter Integration v2.3	VMware vCenter v6.0, 6.5
VMware vROps Integration v2.3	VMware vROps v6.6
VMware vSphere Integration v2.4	VMware vSphere v6.0, 6.5
VMware vRealize Log Insight Integration v2.4	VMware vRealize Log Insight v4.3
WebSphere MQ Integration v1.12	WebSphere MQ v8
xMatters Integration v1.6	xMatters v5.5
Zabbix Integration v1.0	Zabbix v3.2
Zabbix Polling Integration v3.4	Zabbix v3.2
Zenoss Integration v2.4	Zenoss v4.2

Sizing Recommendations

The sizing recommendations below are guidelines for small, medium and large Cisco Crosswork Situation Manager systems based on input data rate and volume.



In the context of this guide, Managed Devices (MDs) are all of the components in the network infrastructure that generate and emit events:

Small

Environment	CPU	File System
<ul style="list-style-type: none">• 1000 to 5000 Managed Devices (MDs)• Less than 20 users• Up to 5 integrations• Less than 20 Alerts per second	<ul style="list-style-type: none">• 8 Cores• 32GB RAM• 2 x 1GB Ethernet• Physical or Virtual Server	<ul style="list-style-type: none">• 1 TB Local or SAN <p>See retention policy.</p>

Medium

Environment	CPU	File System
<ul style="list-style-type: none">• 5000 to 20,000 MDs• Between 20 and 40 users• Between 6 and 10 integrations• Between 20 and 100 Alerts per second	<ul style="list-style-type: none">• 16 Cores• 64GB RAM• 2 x 1GB Ethernet• Physical or Virtual Server	<ul style="list-style-type: none">• 1 TB Local or SAN <p>See retention policy.</p>

Large

Environment	CPU	File System
<ul style="list-style-type: none">• More than 20,000 MDs• More than 40 users• More than 10 integrations• More than 100 Alerts per second	<ul style="list-style-type: none">• 24+ Cores• 128GB RAM• 2 x 1GB Ethernet• Physical or Virtual Server	<ul style="list-style-type: none">• 1 TB Local or SAN <p>See retention policy.</p>

Virtualization Restrictions

Consider the following restrictions for virtual environments:

- Ideally all Moog servers (guests) should be on the same compute node (host) sharing a hypervisor or virtual machine monitor. This minimizes latency between Moog guests.
- If servers are liable to automated resource balancing (e.g. vMotion) and liable to move compute nodes, then all Moog servers should be moved at the same time. If



this is not possible, then Moog servers should be constrained to movements that minimize the resulting network distance.

- If Moog servers are distributed amongst compute nodes then the network “distance” (logical hops) between the nodes should be minimized.
- Network latency between components may affect Event processing throughput. This is especially true of the core to db servers.

Shared Storage

On any shared compute platform Cisco makes the following recommendations:

- The minimum resource requirements are multiplied by at least 33% to account for shared resource usage and allocation.
- Storage latency will reduce effective throughput at the core processing layer and should be minimised within the available constraints of a SAN.
- Cisco Crosswork Situation Manager should be treated as a highly transactional system and not placed on the same compute node as other highly transactional applications that may cause SAN resource contention.
- SAN port and array port contention should be minimized
- Storage medium should be as fast as possible to minimize the transaction times to the database.

Retention Policy

You can calculate the amount of disk space in GB required for the database server using the following calculation:

$$(es \times eps \times d \times 86,400) \times 1.2 / 1,000,000$$

For this calculation: es = average event size in KB, eps = average events per second, d = number of days of retention and 86,400 represents the number of seconds per day.

For the majority of event sources, you can reasonably estimate a 2KB event size. However, some sources have larger than average events. For example, Microsoft SCOM. A 2KB base takes account of the other event and alert based storage such as an alert's Situation membership and Situation room thread sizes.

The average event rate is across all LAMs and integrations.

Note

If you do not enable the Archiver tool, the historic database will grow indefinitely. See [Archive Situations and Alerts](#) for more information.

For example, the following calculation represents a 400 day retention period with an average event size of 2KB at 300 events per second:



$(2 \times 300 \times 400 \times 86,400) \times 1.2 / 1,000,000 = 24,883.2 \text{ GB}.$

Install and Upgrade Cisco Crosswork Situation Manager

Use this guide to learn how to install Cisco Crosswork Situation Manager:

If you are installing another version, see [Welcome to the Cisco Docs!](#) for more information. Refer to the following topics to help choose the right environment for your Cisco Crosswork Situation Manager deployment:

- The [Cisco Crosswork Situation Manager 7.2.0 Supported Environments](#) topic details supported operating systems and system requirements.
- The [Sizing Recommendations](#) will help you make sure you select hardware to support your data ingestion and user requirements.

If you are upgrading Cisco Crosswork Situation Manager, see [Upgrade - RPM Deployments](#), or [Upgrade - Tarball Deployments](#) as appropriate.

Cisco Crosswork Situation Manager Packages

A Cisco Crosswork Situation Manager deployment comprises several packages. The different implementation procedures offer you flexible ways to deploy the packages, which consist of the following:

- LAMs (integrations) that listen or poll for events, parse and encode them into discrete events, and then write the discrete events to the Message Bus.
- The Message Bus (RabbitMQ) that receives published messages from integrations. It publishes messages destined for data processing (Moogfarmd) and the web application server.
- The system datastore (MySQL) that handles transactional data from other parts of the system: LAMs (integrations), data processing, and the web application server.
- The data processing component (Moogfarmd), an application that consumes messages from the Message Bus. It processes event data in a series of servlet-like modules called Moolets. Moogfarmd reads and writes to the database and publishes messages to the bus.
- The web application server (Tomcat) that reads and writes to the Message Bus and the database.
- A proxy (Nginx) for the web application server and for integrations.
- The search engine (Elasticsearch) for the UI that indexes documents from the indexer Moolet in the data processing series. It returns search results to Tomcat.

The diagram below shows the general data flow between components:



Deployment Options

You have the option to install all Cisco Crosswork Situation Manager packages on a single machine. However, the modular approach of the Cisco Crosswork Situation Manager distribution means fewer dependencies between individual packages. This means you have the flexibility to install different components to different machines. For example, you can install the web server components, Ngix and Tomcat, and the database component, MySQL, on different hosts.

- To quickly deploy Cisco Crosswork Situation Manager for evaluation or learning purposes, you can request [trial access to a SaaS environment](#).
- For smaller deployments, you can run all the components in on a single machine.
 - If you have root access to the machine and want to use yum to install, see [Install Cisco Crosswork Situation Manager on a Single Host with RPM](#).
 - If you do not have root access to the machine where you are installing and you want more control over where you install Cisco Crosswork Situation Manager, see [Single Host Tarball Installation](#).
- For larger deployments, you may install different components to different machines in order to distribute the workload. See [Distributed Installation - RPM](#) or [Distributed Tarball Installation](#).

AWS Marketplace Installation

You can install an instance of the latest version of Cisco Crosswork Situation Manager from the Amazon Web Services (AWS) Marketplace.



The screenshot shows the AWS Marketplace page for Moogsoft AIOps (BYOL). The header includes the AWS Marketplace logo, a search bar, and links to sign in or create an account. The main content area features the Moogsoft logo, the product name, and a description. Below this is a table with details like Customer Rating, Latest Version, Operating System, Delivery Method, Support, and AWS Services Required. To the right, there is a 'Continue' button and a 'Pricing Information' section with a region dropdown.

Customer Rating	***** (0 Customer Reviews)
Latest Version	Moogsoft AIOps Version 6.3.0.beta
Operating System	Linux/Unix, CentOS 7
Delivery Method	64-bit Amazon Machine Image (AMI) (Read more)
Support	See details below
AWS Services Required	Amazon EC2, Amazon EBS

Continue You will have an opportunity to review your order before launching or being charged.

Pricing Information
Use the Region dropdown selector to see software and infrastructure pricing information for the chosen AWS region.

For Region
Asia Pacific (Mumbai)

For more information see the [AWS Marketplace listing](#).

Load Data into AWS Instance

You can start loading data into your AWS Marketplace instance by configuring SSL and setting a password for the limited user account.

Configure SSL

Cisco Crosswork Situation Manager includes a self-signed certificate by default. To remove the browser warning, you can add your own certificate as follows:

1. Replace the contents of `/etc/nginx/ssl/certificate.pem` and `/etc/nginx/ssl/certificate.key` with a self-signed or valid certificate.
2. Restart Nginx (`systemctl restart nginx`) or alternatively offload the TLS certificate to an AWS application load balancer.

Note

To SSH into AWS instance, the default password is `centos`.

Set Password for Tool Runner

Many of the Cisco Crosswork Situation Manager integrations run as a limited user account for security reasons. The user account `moogtoolrunner` needs a password before any data is loaded and the services are restarted. For more information see [Tool Runner](#).

The entire step can be achieved using the following command in an SSH terminal:

```
export PASSWORD=<set a password here>; bash -c "$(curl https://s3.moogsoft.com/downloads/moogsetup.sh)"
```

If changing the password was successful, the following message appears:



```
% Total      % Received % Xferd  Average Speed   Time    Time       Time
Current
                                Dload  Upload   Total     Spent     Left
Speed
100 1170 100 1170    0    0 1105      0 0:00:01 0:00:01 --:--:--
1106
Changing password for user moogtoolrunner.
passwd: all authentication tokens updated successfully.
Match User moogtoolrunner
PasswordAuthentication yes
```

Note

The default username for your AWS instance is 'Admin' and the password is your ECS instance ID, e.g. "i-0dd3a563981f57341". These can be changed or you can create a new login when you first sign in.

Post Install Validation

After an installation or an upgrade it is important to run the relevant validator script (after the required install/upgrade steps have been followed and init scripts run, etc.).

Main Cisco Crosswork Situation Manager server validation (Moogfarmd, LAMs, etc)

```
$MOOGSOFT_HOME/bin/utils/moog_install_validator.sh
```

Errors from the moog_install_validator.sh referencing Pak_docs.html can be ignored.

Apache Tomcat webapp validation

```
$MOOGSOFT_HOME/bin/utils/tomcat_install_validator.sh
```

MySQL schema validation

```
$MOOGSOFT_HOME/bin/utils/moog_db_validator.sh
```

If you have any issues see [Troubleshooting](#).

Go to [Getting Started](#) in the Operator Guide.

Encrypt Database Communications

You can enable SSL to encrypt communications between all Cisco Crosswork Situation Manager components and the MySQL database.

For information on creating SSL keys and certificates for MySQL, see [Creating SSL and RSA Certificates and Keys using MySQL](#).

Establish Trust for the MySQL Certificate

To establish trust for the MySQL database certificate, create a truststore to house the root certificate for the Certificate Authority that signed the MySQL Server certificate.

1. If you upgraded from a previous version of Cisco Crosswork Situation Manager, run the following command to extract the certificate for the root CA for MySQL:



```
mysql_ssl_rsa_setup
```

The command generates new keys and writes them to the /var/lib/mysql directory.

2. Run the java keytool command to create a trust store containing the certificate for the root CA for MySQL.

```
keytool -import -alias mysqlServerCACert -file /var/lib/mysql/ca.pem -  
keystore $MOOGSOFT_HOME/etc/truststore
```

- When keytool prompts you, enter a password for the keystore. You will need this password to configure Cisco Crosswork Situation Manager.
- Answer 'yes' to "Trust this certificate."

Keytool creates a truststore at the path \$MOOGSOFT_HOME/etc/truststore.

Configure Cisco Crosswork Situation Manager to use SSL for Database Communications

After you have created the truststore, edit the Cisco Crosswork Situation Manager configuration to enable SSL.

1. Edit \$MOOGSOFT_HOME/config/system.conf.
2. Inside the MySQL property, uncomment the SSL property and the properties that comprise it. Make sure to uncomment the opening "{" and closing braces "}". For example:

```
, "ssl" :  
{  
# # The location of the SSL truststore.  
# #  
# # Relative pathing can be used, i.e. '.' to mean current directory,  
# # '../truststore' or '../../truststore' etc. If neither relative  
# # nor absolute (using '/') path is used then $MOOGSOFT_HOME is  
# # prepended to it.  
# # i.e. "config/truststore" becomes "$MOOGSOFT_HOME/config/truststore"  
# #  
# #  
# # Specify the server certificate.  
# #  
"trustStorePath" : "etc/truststore",  
  
# "trustStoreEncryptedPassword" :  
"vQj7/yom7e5ensSEb10v2Rb/pgkaPK/40cU1EjYNtQU=",  
  
"trustStorePassword" : "moogsoft"  
}
```

3. Provide the path to the truststore you created. For example:



```
"trustStorePath" : "etc/truststore",
```

4. Edit the password for the truststore. For example:

```
"trustStorePassword" : "moogsoft"
```

See [Moog Encryptor](#) if you want to use an encrypted password. Uncomment `trustStoreEncryptedPassword` and provide the encrypted password for the value. For example:

```
"trustStoreEncryptedPassword" :  
"vQj7/yom7e5ensSEb10v2Rb/pgkaPK/40cU1EjYntQU="
```

5. Save your changes and restart the following components:

- Moogfarmd
- Apache Tomcat
- All LAMs

After you restart, all Cisco Crosswork Situation Manager components encrypt communications with the MySQL database.

[Change Passwords for Default Users](#)

Cisco Crosswork Situation Manager creates systems users for Linux, MySQL, and RabbitMQ during the installation process. As a security measure, you can change the password for these users. After you change the passwords for certain users, you must update Cisco Crosswork Situation Manager configuration to use the new password.

If you run in a distributed environment, you can set unique passwords for all components on each host. Update the configuration files for a host to contain the password for user for the host.

Cisco recommends you encrypt passwords for use in Cisco Crosswork Situation Manager configuration files. See [Moog Encryptor](#). In distributed or high availability environments, encrypt the password using the Moog Encryptor on each machine.

[Linux Users](#)

The Cisco Crosswork Situation Manager installation package creates the following Linux users with login privileges:

- moogsoft
- moogadmin
- rabbitmq
- tomcat
- moogtoolrunner



Execute the `passwd` command to change the password Linux users. For example, to change the password for `moogtoolrunner`:

```
passwd moogtoolrunner
```

Update Configuration

After you change the password for `moogtoolrunner`, update its password in `$MOOGSOFT_HOME/config/servlets.conf`. For example:

```
# The toolrunner user password.  
    # Use either toolrunnerpassword or toolrunnerpassword.  
    toolrunnerpassword: "MyNewPassword"  
    # encrypted_toolrunnerpassword:  
    "rmW2daCwMyI8JGZygfEJj0MZdbIkUqX3tT/OIVfMGyI=",
```

Restart Apache Tomcat to apply the configuration change.

```
service apache-tomcat restart
```

Note

You do not need to update the Cisco Crosswork Situation Manager configuration after you change the password for other Linux users with login privileges.

Other Linux Users

The Cisco Crosswork Situation Manager installation package creates the following users without login privileges:

- `elasticsearch`
- `mysql`
- `nginx`

MySQL User

The Cisco Crosswork Situation Manager installation process creates a user to log in to MySQL: `ermintrude`. Execute the MySQL `SET PASSWORD` statement for the user `ermintrude` to change its password:

```
SET password FOR 'ermintrude'@'localhost' = PASSWORD('<new-password>');
```

If you are running in a distributed environment, update the password for the `ermintrude` user on all other nodes:

```
SET password FOR 'ermintrude'@'<host>' = PASSWORD('<new-password>');
```

Where `<host>` is the name of the remote server connecting to the database. For example, if you have the Cisco Crosswork Situation Manager UI web components running on a separate host named `"mywebserver"`:

```
SET password FOR 'ermintrude'@'mywebserver' = PASSWORD('MyNewPassword');
```



After you change the password for ermintrude, grant it privileges on all the objects in the moogdb and moog_reference databases. For example:

```
GRANT ALL ON moogdb.* TO ermintrude@'localhost' IDENTIFIED BY '<new-  
password>';  
GRANT ALL ON moog_reference.* TO ermintrude@'localhost' IDENTIFIED BY  
'<new-password>';
```

If you are running in a distributed environment, you must grant permissions for the ermintrude user on all nodes. For example:

```
GRANT ALL ON moogdb.* TO ermintrude@'<host>' IDENTIFIED BY '<new-  
password>';  
GRANT ALL ON moog_reference.* TO ermintrude@'<host>' IDENTIFIED BY '<new-  
password>';
```

Where <host> is the name of the remote server connecting to the database. For example, if you have the Cisco Crosswork Situation Manager UI web components running on a separate host named "mywebserver":

```
GRANT ALL ON moogdb.* TO ermintrude@'my' IDENTIFIED BY 'MyNewPassword';  
GRANT ALL ON moog_reference.* TO ermintrude@'<host>' IDENTIFIED BY  
'MyNewPassword';
```

Update Configuration

After you change the password for ermintrude, update its password in \$MOOGSOFT_HOME/config/system.conf. For example:

```
"mysql" :  
  {  
    "host" : "localhost",  
    # The name of the moogdb database  
    "moogdb_database_name" : "moogdb",  
    # The name of the moog_reference database  
    "referencedb_database_name" : "moog_reference",  
    "username" : "ermintrude",  
    #  
    "encrypted_password": "vQj7/yom7e5ensSEb10v2Rb/pgkaPK/40cU1EjYNtQU=",  
    "password" : "MyNewPassword",
```

If you are running in a distributed environment, update the password configuration on every host.

RabbitMQ User

The Cisco Crosswork Situation Manager installation process creates a RabbitMQ user called moogsoft. Execute the rabbitmqctl change_password command to change the password for moogsoft. For example:

```
rabbitmqctl change_password moogsoft <new-password>
```



Update Configuration

After you change the password for moogsoft, update its password in `$MOOGSOFT_HOME/config/system.conf`. For example:

```
# By default the moogsoft username and password are used.
# This needs to match the MooMS broker configuration. If
# commented out a default "guest" user will be used.
#
"username"          : "moogsoft",
"password"          : "MyNewPassword",
# "encrypted_password" :
"e5u00LY3HQJZCltG/caUnVbxVN4hImm4gIOpb4rwpF4=",
```

If you are running in a distributed environment, update the password configuration on every host.

Scale Your Cisco Crosswork Situation Manager Implementation

Cisco Crosswork Situation Manager supports several options to help you scale your implementation to meet your performance needs. Use the [built-in diagnostic tools](#) to monitor your system for signs that it is time to scale.

For information on the performance tuning capabilities of individual Cisco Crosswork Situation Manager components, see [Moogsoft AIOps Component Performance](#).

Horizontal Scaling

Cisco Crosswork Situation Manager currently supports horizontal scaling at the integration (LAM) and visualization (Ngnix + Tomcat) layers.

- You can add more LAMs, either on additional servers or on the same server, to achieve higher event rates. In this case, you have the option to configure event sources to send to the parallel LAMs separately or to implement a load balancer in front of the LAMs.
- You can add Ngnix/Tomcat UI "stacks" behind a load balancer to increase performance for UI users. Adding UI stacks does not always provide better performance. It can degrade performance by adding more connection pressure to the database.

The following are typical horizontal scaling scenarios:

- You can add an additional LAM to process incoming events if you see that, despite attempts to tune the number of threads for an individual LAM, its event rate hits a plateau. This is a sign that the LAM is the bottleneck, so adding other instances of the LAM behind a load balancer will allow a higher event processing rate.
- You can add an additional UI stack if database pool diagnostics for Tomcat suggest that all or most of the database connections are constantly busy with long running connections, but the database itself is performing fine.



The data processing layer (moogfarmd) is not currently well suited to horizontal scaling. Moolets of the same type cannot currently share processing. Adding more Moolets like the AlertBuilder in an attempt to increase the event processing rate is likely to lead to database problems.

Vertical Scaling

All Cisco Crosswork Situation Manager components ultimately benefit from being run on the best available hardware, but the data processing layer (moogfarmd) benefits most from this approach. Depending on the number and complexity of Moolets in your configuration, you will see performance benefits in data processing on servers having the fastest CPUs with numerous cores and a large amount of memory. This enables you to increase the number of threads for moogfarmd to improve processing speed. You should also locate the database on the most feasibly powerful server (clock speed, number of cores and memory) with the biggest/fastest disk.

Distributed Installations

In some cases you distribute Cisco Crosswork Situation Manager components among different hosts to gain performance because it reduces resource contention on a single server: The most common distribution is to install the database on a separate server, ideally within the same fast network to minimize risk of latency. An additional benefit of this move is that it allows you to run a clustered or master/slave database for redundancy.

Another common distribution is to install the UI stack (Nginx) on a separate server within the same fast network.

Some integrations (LAMs) benefit in being closer to the source so are a candidates for distribution. You can also break the data processing layer (moogfarmd) into its component Moolets and installed across separate servers. This gives each moogfarmd process a separate database pool which may provide performance benefits in certain situations.

See [Distributed Installation - RPM](#) for more information.

Uninstall Cisco Crosswork Situation Manager

Follow the instructions in this topic if you need to uninstall Cisco Crosswork Situation Manager and its supporting packages.

Be sure to backup any files that you may need again for another installation.

Stop Core Cisco Crosswork Situation Manager and Supporting Services

1. Stop all core Cisco Crosswork Situation Manager services:

```
service moogfarmd stop
service logfilelamd stop
service restlamd stop
service socketlamd stop
service trapdlamd stop
```



2. Stop any additional moog_farmd or lam instances running as services.

```
service <service name> stop
```

3. As a precaution, forcibly kill any remaining core Cisco Crosswork Situation Manager processes:

```
kill -9 $(ps -ef|grep java|grep lam|awk '{print $2}') 2>/dev/null
kill -9 $(ps -ef|grep java|grep moog_farmd|awk '{print $2}') 2>/dev/null
```

4. Stop all supporting services:

```
service nginx stop
service elasticsearch stop
service apache-tomcat stop
service mysqld stop
service rabbitmq-server stop
```

Uninstall Core Cisco Crosswork Situation Manager Packages and Remove Directories and Users

1. Uninstall Core Cisco Crosswork Situation Manager Packages

```
yum remove $(rpm -qa|grep moogsoft)
```

If the above command produces errors such as 'Error in PREUN scriptlet' then the following command can be run to bypass script errors:

```
yum -y --setopt=tsflags=noscripts remove $(rpm -qa|grep moogsoft)
```

2. Remove Core Cisco Crosswork Situation Manager Directories

```
rm -rf /usr/share/moogsoft
rm -rf /var/lib/moogsoft
rm -rf /var/log/moogsoft
rm -rf /var/run/moogsoft
```

3. Remove any Cisco crontab entries with this command:

```
crontab -l | egrep -v "moog|JAVA_HOME" | crontab -
```

4. Remove Cisco system users (and their home directories):

```
userdel -r moogsoft
userdel -r moogadmin
userdel -r moogtoolrunner
groupdel moogsoft
```

Uninstall Supporting Applications

Follow these steps to remove the supporting applications Apache-Tomcat, ElasticSearch, MySQL, Nginx and RabbitMQ.

Uninstalling Apache Tomcat

**Note**

Assumption: The Apache Tomcat service has already been stopped as per previous instructions above

To uninstall Apache Tomcat remove the installation directories and the service script:

Note

Please note: Apache Tomcat is not actually installed as an rpm package but is deployed as a tarball (via the moog_init_ui.sh script).

```
rm -rf /usr/share/apache-tomcat
rm -rf /var/run/apache-tomcat
rm -f /etc/init.d/apache-tomcat
```

To remove the tomcat system user and its home directory:

```
userdel -r tomcat
```

Uninstalling Elasticsearch

Note

Assumption: The Elasticsearch service has already been stopped as per previous instructions above.

To remove the Elasticsearch package:

```
yum remove elasticsearch
```

To remove related directories run the following commands:

```
rm -rf /usr/share/elasticsearch
rm -rf /var/lib/elasticsearch
```

Uninstalling MySQL

Note

Assumption: The mysqld service has already been stopped as per previous instructions above.

Remove the MySQL community packages with the following command:

```
yum remove $(rpm -qa|grep mysql)
```

To remove the related directories:

```
rm -rf /usr/share/mysql
rm -rf /var/lib/mysql
```

To remove the MySQL system user and its home directory and group:

```
userdel -r mysql
```



Uninstalling Nginx

Note

Assumption: The Nginx service has already been stopped as per previous instructions above.

Remove the nginx and supporting packages with the following command:

```
yum remove nginx
```

To remove related directories:

```
rm -rf /etc/nginx
rm -rf /usr/lib64/nginx
rm -rf /usr/share/nginx
rm -rf /var/log/nginx
rm -rf /var/lib/nginx
```

To remove the Nginx system user and its home directory and group:

```
userdel -r nginx
```

Uninstalling RabbitMQ

Note

Assumption: RabbitMQ server service has already been stopped as per previous instructions above.

Remove the rabbitmq-server package with the following command:

```
yum remove rabbitmq-server
```

To remove related directories:

```
rm -rf /etc/rabbitmq
rm -rf /usr/lib/ocf/resource.d/rabbitmq
rm -rf /var/log/rabbitmq
rm -rf /var/lib/rabbitmq
```

To stop the erlang epmd daemon:

```
epmd -kill
```

Note

Please note: The above command may not be necessary on EL7 installs.

To remove the RabbitMQ system user and its home directory and group:

```
userdel -r rabbitmq
```



Uninstall Remaining Packages and Remove Yum Repositories

Optionally, follow these steps to remove the remaining packages that are typically added during a Cisco Crosswork Situation Manager installation and clean up the Yum repositories:

Remove remaining packages:

Warning

Important: Note that the below list of packages is based on reverting back to a "minimal" installation of CentOS 6.9 and will vary with different versions of Linux and installation level.

Care should be taken not to remove packages that impact other important applications that may be installed on the server.

Review carefully the yum summary before proceeding with the removal - specifically any other packages listed in the "Removing for dependencies:" output.

In the list of removal packages below, the **libX*** and **perl*** packages may typically impact other applications.

To remove the remaining packages, run these commands:

```
yum remove GeoIP GeoIP-GeoLite-data GeoIP-GeoLite-data-extra \
apr compat-readline5 erlang fontconfig freetype gd geoipupdate jdk1.8.0_121 \
libX11 libX11-common libXau libXpm libgfortran libjpeg-turbo libkqueue
libpng libxcb libxslt \
nginxfilesystem \
perl perl-DBI perl-Module-Pluggable perl-Pod-Escapes perl-Pod-Simple perl-
libs perl-version \
socat tomcat-native
```

Remove Yum Repositories:

Remove EPEL Yum Repository:

```
yum remove epel-release
rm -f /etc/yum.repos.d/epel*
```

Remove MySQL Community Yum Repository:

```
yum remove mysql-community-release
rm -f /etc/yum.repos.d/mysql*
```

Remove remaining Yum Repositories:

```
rm -f /etc/yum.repos.d/elasticsearch.repo
rm -f /etc/yum.repos.d/moog.repo
rm -f /etc/yum.repos.d/rabbitmq_rabbitmq-server.repo
```

Control Cisco Crosswork Situation Manager Processes

This topic describes the commands for starting, stopping or restarting individual Cisco Crosswork Situation Manager processes.



To configure process startup when Cisco Crosswork Situation Manager fails or restarts see [Configure Services to Restart](#).

Dependencies

Integrations (LAMs), Moogfarmd, and Tomcat depend on the system processes: MySQL RabbitMQ, Nginx, and Elasticsearch. So when starting Cisco Crosswork Situation Manager processes:

1. Start or verify the following are started:
 - MySQL
 - RabbbitMQ
 - Nginx
 - Elasticsearch
2. Start or restart integrations (LAMs), Moogfarmd, or Tomcat.

Similarly, if you plan to stop any one of MySQL, RabbitMQ, Nginx, or Elasticsearch, stop integrations (LAMs), Moogfarmd, and Tomcat first.

Init Scripts for RPM Installs

If you performed an RPM installation as root, use the service init script to start and stop Cisco Crosswork Situation Manager processes:

```
service <service-name> start|stop|restart
```

The service names are as follows:

- MySQL: mysqld
- RabbitMQ: rabbitmq-server
- Nginx: nginx
- Elasticsearch: elasticsearch
- Tomcat: apache-tomcat
- moogfarmd
- For LAMs , refer to the individual LAM references for the service names.

For more information, see the documentation on managing system services for your operating system.

Process Control for Non-root Installations

For customers who follow the [Single Host Installation for Non-root Users](#) procedure, Cisco Crosswork Situation Manager includes a process control utility to let you:



- Start a process
- Stop a process
- Restart a process
- Check the status, running or stopped, of a process.

The process control utility resides at `$MOOGSOFT_HOME/bin/utils/process_cntl`.

When you [install Moogsoft AIOps as a user other than root](#), you choose a user to run the installation and initialize the system. Use the same user credentials when controlling Cisco Crosswork Situation Manager components to ensure that you have the proper permissions and access.

Process Control Utility Reference

The Process Control utility uses the following syntax:

```
process_cntl [ [--process_name] <name> ] [--loglevel] <loglevel> ] [--  
service_instance <instance> ] {start|stop|status|restart|help} ]
```

The arguments for the utility are as follows:

Argument	Input	Description
-h, --help	-	Display the process_cntl syntax and option descriptions.
--loglevel	DEBUG INFO WARN	Log level controlling the amount of information that process control logs. Defaults to WARN. This flag only works for Cisco Crosswork Situation Manager processes.
--process_name	process name	<p>The name of the process to control. You can specify one of the core processes:</p> <ul style="list-style-type: none">• rabbitmq - RabbitMQ message broker• mysql - MySQL database• nginx - Nginx web server.• elasticsearch - Elasticsearch search engine.• apache-tomcat - Apache Tomcat servlet container.• moog_farmd - Cisco Crosswork Situation Manager event processing process. <p>Alternatively specify an integration (LAM). Run <code>process_cntl -h</code> for a full list of integrations and</p>



syntax.

See [Implementer Guide](#) for a brief description of the packages.

-- service_instance	instance name	The name of the process instance if there is more than one running on the system.
start	-	Start a stopped process.
stop	-	Stop a running process.
status	-	Display the status of the process: running or stopped.
restart	-	Restart a running process.

RPM Installation and Upgrade

You can only perform an RPM installation and upgrade if you have root access to the RHEL / Centos 7 system where you are installing Cisco Crosswork Situation Manager.

The standard RPM installation requires you to use the yum package-management utility. If you have root access but you cannot use yum to connect to package repositories from outside your network, you can perform an offline RPM installation.

To install using RPMs, select the pre-install steps that suit your requirements:

If you cannot install Cisco Crosswork Situation Manager as the root user, see [Tarball Installation and Upgrade](#). For other installation and upgrade options see [Install and Upgrade Moogsoft AIOps](#).

Pre-Install Cisco Crosswork Situation Manager - Offline RPMs

If you cannot use Yum to connect to package repositories outside your network from the machines where you are installing Cisco Crosswork Situation Manager, you can download tarball packages of the repositories to run an offline installation. The offline repository distributions include all required packages to install Cisco Crosswork Situation Manager on a RHEL/CentOS 7 server.

These instructions guide you through the process to set up the local Yum repositories so you can continue with an offline installation or upgrade. After you set up your repositories, you can continue with one of the following:

- [Single Host Installation](#)
- [Distributed Installation - RPM](#)
- [Upgrade AIOps](#)

This procedure does not support a relocatable installation.

Before You Begin



- Ensure you have root access to the system where you are installing Cisco Crosswork Situation Manager.

Cisco Crosswork Situation Manager Installation Files

To improve download times, the distribution for the offline installation of Cisco Crosswork Situation Manager comes in two separate archives:

- A "BASE" repository containing the dependent packages to install Cisco Crosswork Situation Manager for RHEL/CentOS 7. The base package follows the following naming convention:

`<date/timestamp>-MoogsoftBASE7_offline_repo.tar.gz`

Example: `2018-09-26-1537962719-MoogsoftBASE7_offline_repo.tar.gz`

- An "ESR" repository that contains the standard Cisco Crosswork Situation Manager RPMs and ancillary packages (Tomcat, RabbitMQ, JRE etc)

ESR: `<date/timestamp>-MoogsoftESR_<version>_offline_repo.tar.gz`

Example: `2018-09-26-1537962719-MoogsoftESR_7.2.0_offline_repo.tar.gz`

Download Installation Files

Before you can set up the local Yum repositories, you need to download installation files from a machine connected to the internet. Then copy the installation files to a directory on the target system. The examples use /home for the installation file directory.

1. Download the 'ESR' and 'BASE' RPM tarballs from the 'speedy' server using the following two links in internet browser on an internet-connected host. Provide your 'speedy' access credentials when prompted by the browser.
 - a. https://speedy.moogsoft.com/offline/aiops/2019-04-24-1556146392-MoogsoftBASE7_offline_repo.tar
 - b. https://speedy.moogsoft.com/offline/aiops/2019-04-24-1556146392-MoogsoftESR_7.2.0_offline_repo.tar
2. Copy the two downloaded tarballs to the target offline system.
3. Download the MySQL-Community packages on an internet-connected host and copy them to the target offline system.

For example on RHEL7/CentOS7:

```
curl -L -O https://repo.mysql.com/yum/mysql-5.7-community/el/7/x86_64/mysql-community-libs-5.7.22-1.el7.x86_64.rpm
curl -L -O https://repo.mysql.com/yum/mysql-5.7-community/el/7/x86_64/mysql-community-libs-compat-5.7.22-1.el7.x86_64.rpm
curl -L -O https://repo.mysql.com/yum/mysql-5.7-community/el/7/x86_64/mysql-community-server-5.7.22-1.el7.x86_64.rpm
curl -L -O https://repo.mysql.com/yum/mysql-5.7-
```



```
community/el/7/x86_64/mysql-community-common-5.7.22-1.el7.x86_64.rpm
curl -L -O https://repo.mysql.com/yum/mysql-5.7-
community/el/7/x86_64/mysql-community-client-5.7.22-1.el7.x86_64.rpm
```

Prepare the Local Yum Repositories

The following procedure describes how to create local Yum repositories to house the installation packages for Cisco Crosswork Situation Manager that you downloaded. If you are running a distributed installation, follow this procedure on each machine where you run Cisco Crosswork Situation Manager components.

1. Create directories that will house the repositories. For example:

```
sudo mkdir -p /media/localRPM/BASE/
```

```
sudo mkdir -p /media/localRPM/ESR/
```

2. Extract the two packages into separate directories. For example:

```
tar xzf *-MoogsoftBASE7_offline_repo.tar.gz -C /media/localRPM/BASE/
```

```
tar xzf *-MoogsoftESR_7.2.0_offline_repo.tar.gz -C /media/localRPM/ESR/
```

3. Move the existing /etc/yum.repos.d. directory to create a backup:

```
mv /etc/yum.repos.d /etc/yum.repos.d-backup
```

4. Creating an empty /etc/yum.repos.d directory

```
mkdir /etc/yum.repos.d
```

5. Create a local.repo for Yum:

```
vi /etc/yum.repos.d/local.repo
```

6. Edit the contents of local.repo. Verify the path for the base and for the baseurl points to the directories you created.

```
[BASE]
name=MoogCentOS-$releasever - MoogRPM
baseurl=file:///media/localRPM/BASE/RHEL
gpgcheck=0
enabled=1
[ESR]
name=MoogCentOS-$releasever - MoogRPM
baseurl=file:///media/localRPM/ESR/RHEL
gpgcheck=0
enabled=1
```

7. Clean the Yum cache to remove cached files from any enabled repository:

```
yum clean all
```



8. Verify Yum detects the the newly created local repositories:

```
yum info "moogsoft-*
```

Available Packages

```
Arch      : x86_64
Version   : 7.2.0
Release   : 8
Size      : 76 M
Repo      : ESR
Summary   : Algorithmic Intelligence for IT Operations
URL       : https://www.moogsoft.com
License   : Proprietary
Description : Moogsoft AIOps (7.2.0) - Build: 142 - (Revision:
              : 7a2df5e47e10ee17ab646bf5d06f3b31437e3bac)
```

The results should include the following packages:

```
Name      : moogsoft-db
Name      : moogsoft-integrations
Name      : moogsoft-integrations-ui
Name      : moogsoft-mooms
Name      : moogsoft-search
Name      : moogsoft-server
Name      : moogsoft-ui
Name      : moogsoft-utils
Name      : moogsoft-common
```

9. Update the system if needed. Create an [exclusion list](#) if you prefer to keep certain packages away from update:

```
yum update
```

10. Install the MySQL RPMs:

```
yum install mysql-community-*5.7.22*.rpm
```

If Yum reports errors regarding mariadb, replace mariadb with the equivalent MySQL package. The following script removes mariadb-server and replaces it with mysql-community-server:

```
echo "remove mariadb-server" >> /tmp/mysql.libs
echo "install mysql-community-libs-compat-5.7.22" >> /tmp/mysql.libs
echo "install mysql-community-server-5.7.22" >> /tmp/mysql.libs
echo "run" >> /tmp/mysql.libs
cat /tmp/mysql.libs | yum shell -y
```

Your local Yum repos are ready. Now you can proceed with your offline install or upgrade. See the following topics for more information:

- [Install Cisco Crosswork Situation Manager on a Single Host with RPM](#)
- [Distributed Installation - RPM](#)



- [Upgrade AIOps](#)

[Install Cisco Crosswork Situation Manager on a Single Host with RPM](#)

This topic guides you through the installation process for Cisco Crosswork Situation Manager using RPM via Yum.

Before You Begin

Before you install Cisco Crosswork Situation Manager, ensure you have met the following requirements:

- You have root access to the machine on which you are installing Cisco Crosswork Situation Manager.
- You have completed the pre-installation procedures for your environment as outlined in these documents:
 - [Pre-install Moogsoft AIOps](#)
 - [Pre-install Offline RPMs](#)

Install Cisco Crosswork Situation Manager

To install Cisco Crosswork Situation Manager on a single host machine:

1. Download the RPMs and install them:

```
yum -y install moogsoft-db-7.2.0 \
moogsoft-common-7.2.0 \
moogsoft-integrations-7.2.0 \
moogsoft-integrations-ui-7.2.0 \
moogsoft-mooms-7.2.0 \
moogsoft-search-7.2.0 \
moogsoft-server-7.2.0 \
moogsoft-ui-7.2.0 \
moogsoft-utils-7.2.0
```

2. Set the environment variables required for Cisco Crosswork Situation Manager. If you are using the default Bash shell, you can set the variables in the bashrc file:

```
vi ~/.bashrc
```

Add the following by copying and pasting each line, one at a time:

```
export MOOGSOFT_HOME=/usr/share/moogsoft
export JAVA_HOME=/usr/java/latest
export APPSERVER_HOME=/usr/share/apache-tomcat
export PATH=$PATH:$MOOGSOFT_HOME/bin:$MOOGSOFT_HOME/bin/utils
```

3. Start a new Bash shell or source the bashrc file as follows:

```
source ~/.bashrc
```



4. Initialize the installation:

```
$MOOGSOFT_HOME/bin/utils/moog_init.sh -I MY_ZONE -u root
```

The default password for MySQL is empty, so press <Enter> when prompted for the MySQL root password. If you created a separate user in the database for Cisco Crosswork Situation Manager, use the credentials for that user.

Follow the instructions and confirm prompts about the hostname and ports.

5. Start Moogfarmd:

```
service moogfarmd start
```

Verify your installation following the steps in [Post Install Validation](#).

Follow the [Troubleshooting](#) guide to resolve any issues.

Distributed Installation - RPM

You can install the different components of Cisco Crosswork Situation Manager to multiple machines in order to distribute the workload. Dividing the workload for Cisco Crosswork Situation Manager components between multiple machines can provide performance improvement, but it does not provide redundancy or failover. For those features, see [High Availability Overview](#).

Distributed Installation Procedure

1. Choose which host each component should be installed on, and run one or more of the following commands on the selected hosts as required by your distributed architecture.

```
yum install moogsoft-db-7.2.0
yum install moogsoft-integrations-db-7.2.0
yum install moogsoft-integrations-ui-db-7.2.0
yum install moogsoft-mooms-db-7.2.0
yum install moogsoft-search-db-7.2.0
yum install moogsoft-server-db-7.2.0
yum install moogsoft-ui-db-7.2.0
yum install moogsoft-utils-db-7.2.0
yum install moogsoft-common-db-7.2.0
```

2. Set the environment variables in the bashrc file on each host with the following commands:

```
export MOOGSOFT_HOME=/usr/share/moogsoft
export JAVA_HOME=/usr/java/latest
export APPSERVER_HOME=/usr/share/apache-tomcat
export PATH=$PATH:$MOOGSOFT_HOME/bin:$MOOGSOFT_HOME/bin/utils
```

3. Initialize the installation as described below.

Initialize a Distributed Installation



Each installed package has its own init script (for example `moog_init_ui.sh`). The individual package init scripts make configuring relevant properties more straightforward, such as those in `$MOOGSOFT_HOME/config/system.conf`.

For example, `moog_init_ui.sh` includes the following optional arguments:

- `-d` Configure `system.conf` with the hostname and port of the MySQL server
- `-m` Configure `system.conf` with the hostname and port of the RabbitMQ (MooMS) server
- `-s` Configure `system.conf` with the hostname and port of the search Elasticsearch server
- `-z` Configure `system.conf` and other UI files with the Zone name and RabbitMQ vhost

Examples Initializations in Distributed Installations

SERVER1 and **SERVER2** are example hostnames. **MY_ZONE** is an example MooMS Zone name.

Example 1: moogsoft-db on SERVER1 and all other RPMs on SERVER2

The **db** package can be initialized with basic options but most other packages then need to know its location for their initialization.

1. Initialize the database on **SERVER1**:

```
moog_init_db.sh -Iu root
```

2. On **SERVER1**, connect to MySQL as the root user and grant all access on the **moogdb** and **moog_reference** databases to the **ermintrude** user on **SERVER2**:

```
GRANT ALL ON moogdb.* TO ermintrude@'SERVER2' IDENTIFIED BY 'm00';
GRANT ALL ON moog_reference.* TO ermintrude@'SERVER2' IDENTIFIED BY 'm00';
GRANT ALL ON historic_moogdb.* TO ermintrude@'SERVER2' IDENTIFIED BY 'm00';
```

3. Initialize the remaining components on **SERVER2**:

```
moog_init_mooms.sh -pz MY_ZONE
moog_init_lams.sh -bz MY_ZONE -d SERVER1:3306
moog_init_search.sh -d SERVER1:3306
moog_init_server.sh -bz MY_ZONE -d SERVER1:3306
moog_init_ui.sh -twxfz MY_ZONE -d SERVER1:3306
```

Example 2: moogsoft-ui on SERVER1 and all other RPMs on SERVER2

On initialization, the **ui** package needs to know the following:

- The host:port of the database.



- The host:port of the MooMs broker
- The host_port of Elasticsearch
- The host:port of the MooMs admin console
- The MooMs Zone to connect to

1. Initialize the components on **SERVER2**:

```
moog_init_db.sh -Iu root
moog_init_mooms.sh -pz MY_ZONE
moog_init_server.sh -b
moog_init_search.sh
moog_init_lams.sh -bz MY_ZONE
```

2. On **SERVER2**, connect to MySQL as the root user and grant all access on the **moogdb** and **moog_reference** databases to the **ermintrude** user on **SERVER1**:

```
GRANT ALL ON moogdb.* TO ermintrude@'SERVER1' IDENTIFIED BY 'm00';
GRANT ALL ON moog_reference.* TO ermintrude@'SERVER1' IDENTIFIED BY 'm00';
```

3. By default, Elasticsearch listens only on localhost so must be made to listen on the outbound interface. On **SERVER2**, edit the `/etc/elasticsearch/elasticsearch.yml` file and set the following property:

```
http.host: 0.0.0.0
```

4. On **SERVER2**, restart the Elasticsearch service:

```
service elasticsearch restart
```

5. Finally, run this command on **SERVER1**:

```
moog_init_ui.sh -twxfz MY_ZONE -c SERVER2:15672 -d SERVER2:3306 -m
SERVER2:5672 -s SERVER2:9200 -n
```

Example 3: moogsoft-server on SERVER1 and all other RPMs on SERVER2

On initialization, the **server** package needs to know the following:

- The host:port of the database
- The host:port of MooMS broker
- The MooMS Zone to connect to

1. Initialize the components on **SERVER2**:

```
moog_init_db.sh -Iu root
moog_init_mooms.sh -pz MY_ZONE
moog_init_lams.sh -bz MY_ZONE
```



```
moog_init_search.sh
moog_init_ui.sh -twxfz MY_ZONE
```

2. On **SERVER2**, connect to mysql as the root user and grant all access on the **moogdb** and **moog_reference** databases to the **ermintrude** user on **SERVER1**:

```
GRANT ALL ON moogdb.* TO ermintrude@'SERVER1' IDENTIFIED BY 'm00';
GRANT ALL ON moog_reference.* TO ermintrude@'SERVER1' IDENTIFIED BY
'm00';
```

3. Run this command on **SERVER1**:

```
moog_init_server.sh -bez MY_ZONE -d SERVER2:3306 -m SERVER2:5672
```

Note

The Moogfarmd service is not started or stopped by the `init` script so you will need to do this manually.

Example 4: moogsoft-lams on SERVER1 and all other RPMs on SERVER2

On initialization, the **LAM** package needs to know the following:

- The host:port of the database
- The host:port of MooMS broker
- The MooMS Zone to connect to

1. Initialize the components on **SERVER2**:

```
moog_init_db.sh -Iu root
moog_init_mooms.sh -pz MY_ZONE
moog_init_server.sh -bz MY_ZONE
moog_init_search.sh
moog_init_ui.sh -twxfz MY_ZONE
```

2. On **SERVER2**, connect to mysql as the root user and grant all access on the **moogdb** and **moog_reference** databases to the **ermintrude** user on **SERVER1**:

```
GRANT ALL ON moogdb.* TO ermintrude@'SERVER1' IDENTIFIED BY 'm00';
GRANT ALL ON moog_reference.* TO ermintrude@'SERVER1' IDENTIFIED BY
'm00';
```

3. Run this command on **SERVER1**:

```
moog_init_lams.sh -bz MY_ZONE -d SERVER2:3306 -m SERVER2:5672
```

Upgrade a Distributed Installation

Note

No matter how many RPMs have been installed on a single host, you must upgrade all packages on that host at the same time.



You cannot run different versions of packages at the same time in a distributed environment, so upgrade all hosts to the same package version at the same time.

The following example demonstrates the type of error which yum will produce if only one out of two packages is upgraded. If moogsoft-server and moogsoft-integrations have been installed on a single host, running an upgrade for only moogsoft-server (via `yum update moogsoft-server`) will generate errors from yum similar to those below:

Transaction Check Error:

```
file /usr/share/moogsoft/lib/events_analyser.jar from install of
moogsoft-server-7.2.0-436.x86_64 conflicts with file from package moogsoft-
integrations-7.2.0-435.x86_64
file /usr/share/moogsoft/lib/farmd_cntl.jar from install of moogsoft-
server-7.2.0-436.x86_64 conflicts with file from package moogsoft-
integrations-7.2.0-435.x86_64
file /usr/share/moogsoft/lib/ha_cntl.jar from install of moogsoft-server-
7.2.0-436.x86_64 conflicts with file from package moogsoft-integrations-
7.2.0-435.x86_64
file /usr/share/moogsoft/lib/moobot.jar from install of moogsoft-server-
7.2.0-436.x86_64 conflicts with file from package moogsoft-integrations-
7.2.0-435.x86_64
file /usr/share/moogsoft/lib/moog_add_alert_custom_field.jar from install
of moogsoft-server-7.2.0-436.x86_64 conflicts with file from package
moogsoft-integrations-7.2.0-435.x86_64
file /usr/share/moogsoft/lib/moog_add_sitn_custom_field.jar from install
of moogsoft-server-7.2.0-436.x86_64 conflicts with file from package
moogsoft-integrations-7.2.0-435.x86_64
file /usr/share/moogsoft/lib/moog_config_reader.jar from install of
moogsoft-server-7.2.0-436.x86_64 conflicts with file from package moogsoft-
integrations-7.2.0-435.x86_64
file /usr/share/moogsoft/lib/moog_encryptor.jar from install of moogsoft-
server-7.2.0-436.x86_64 conflicts with file from package moogsoft-
integrations-7.2.0-435.x86_64
file /usr/share/moogsoft/lib/moog_farmd.jar from install of moogsoft-
server-7.2.0-436.x86_64 conflicts with file from package moogsoft-
integrations-7.2.0-435.x86_64
...
```

The supported upgrade path (for the above example) runs the following command:

```
yum update moogsoft-server moogsoft-integrations
```

You must run the relevant init scripts so the updated components know where the rest of the packages are located.

Problems?: [Troubleshooting](#)

[Pre-install Cisco Crosswork Situation Manager](#)

Follow these steps to prepare a host for an RPM installation of Cisco Crosswork Situation Manager. You don't need to complete these steps if you are not using the RPM installation.



Before You Begin

Before you begin your installation, verify the following:

- You have root level access to a Centos 7 / RHEL 7 system where you plan to install Cisco Crosswork Situation Manager.
- You have read through the [Moogsoft AIOps 7.2.0 Supported Environments](#).

Install Prerequisites for Cisco Crosswork Situation Manager

Install the prerequisites for Cisco Crosswork Situation Manager as follows:

1. Install the [Extra Packages for Enterprise Linux: EPEL](#) Yum repository. For example:

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

2. Verify the creation of the file `/etc/yum.repos.d/epel.repo`

3. Install the "el7" MySQL Community Yum repository: For example:

```
yum -y install http://repo.mysql.com/mysql57-community-release-el7.rpm
```

Verify the creation of the file `/etc/yum.repos.d/mysql-community.repo`.

4. Install the RabbitMQ Erlang el7 package. For example:

```
yum -y install https://github.com/rabbitmq/erlang-rpm/releases/download/v20.1.7/erlang-20.1.7-1.el7.centos.x86_64.rpm
```

Alternatively you can find the file at <https://github.com/rabbitmq/erlang-rpm/releases/tag/v20.1.7>.

5. Install the RabbitMQ Yum repository: For example:

```
curl -s https://packagecloud.io/install/repositories/rabbitmq/rabbitmq-server/script.rpm.sh | sudo bash
```

6. Verify the creation of the file `/etc/yum.repos.d/rabbitmq_rabbitmq-server.repo`.

7. Install the Elasticsearch public key. For example:

```
rpm --import https://artifacts.elastic.co/GPG-KEY-elasticsearch
```

8. Create the Elasticsearch Yum repository: `/etc/yum.repos.d/elasticsearch.repo` with the following contents:

```
[elasticsearch-5.x]
name=Elasticsearch repository for 5.x packages
baseurl=https://artifacts.elastic.co/packages/5.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
```



```
enabled=1
autorefresh=1
type=rpm-md
```

9. Create a Bash script named `create_nginx_repo.sh` with the following contents:

```
#!/bin/bash

echo '[nginx]' > /etc/yum.repos.d/nginx.repo
echo 'name=nginx repo' >> /etc/yum.repos.d/nginx.repo
echo 'baseurl=http://nginx.org/packages/OS/OSRELEASE/$basearch/' >>
/etc/yum.repos.d/nginx.repo
echo 'gpgcheck=0' >> /etc/yum.repos.d/nginx.repo
echo 'enabled=1' >> /etc/yum.repos.d/nginx.repo

OS_VERSION=$(cat /etc/system-release)
case "$OS_VERSION" in
    CentOS*release\ 7* )
        sed -i -e 's/OS/centos/' -e 's/OSRELEASE/7/'
        /etc/yum.repos.d/nginx.repo;;
    Red\ Hat*release\ 7* )
        sed -i -e 's/OS/rhel/' -e 's/OSRELEASE/7/'
        /etc/yum.repos.d/nginx.repo;;
esac
```

10. Execute `create_nginx_repo.sh` to create the Nginx Yum repo. For example:

```
bash create_nginx_repo.sh
```

11. Verify the NSS and OpenSSL packages are up to date for your system. For example:

```
yum -y update nss openssl
```

12. Create the Moogsoft Yum repository as a new file `/etc/yum.repos.d/moogsoft-aiops.repo` with the following contents:

```
[moogsoft-aiops]
name=moogsoft-aiops-latest
baseurl=https://<login>:<password>@speedy.moogsoft.com/repo/aiops/esr
enabled=1
gpgcheck=0
sslverify=0
```

13. Set SELinux to permissive mode or disable completely. For example to set SELinux to permissive mode:

```
setenforce 0
```

If you want to disable SELinux at boot time, you can edit the file `/etc/sysconfig/selinux`.

Next step: [Install Moogsoft AIOps on a Single Host with RPM](#)



Upgrade - RPM Deployments

This guide provides instruction on how to upgrade to Cisco Crosswork Situation Manager ESR v7.2.0. You must upgrade to v7.1.0 before you upgrade to v7.2.0.

For instructions on how to upgrade from a supported version, refer to the following topics:

- [Upgrade RPM v6.5.x to RPM v7.2.x](#)
- [Upgrade RPM v7.0.x to RPM v7.2.x](#)
- [Upgrade RPM v7.1.x to RPM v7.2.x](#)

For instructions on how to upgrade other releases, see [/document/preview/47128#UUIDcc26a467cc7c84ff916e9c28adb033d0.Releases](#)

Upgrade RPM v6.5.x to RPM v7.2.x

This topic describes the upgrade procedure of RPM deployments from Cisco Crosswork Situation Manager v6.5.x to Cisco Crosswork Situation Manager v7.2.0. Refer to [Upgrade - RPM Deployments](#) for general information and links to upgrades for other versions.

Before You Begin

Before you begin the upgrade process, complete the following prerequisite steps as appropriate.

Prepare for an Offline Upgrade

To prepare for an offline upgrade, where the Cisco Crosswork Situation Manager packages reside in a 'local' Yum repository, follow steps 1-8 in section "Prepare the Local yum Repositories" in the [Pre-Install Moogsoft AIOps - Offline RPMs](#) guide. Then follow this guide to perform the upgrade. Take into account any step-specific notes regarding offline upgrades.

Note UI Integrations Configuration

If you are using any UI integrations, make a note of their connection and configuration details before you begin the upgrade. You will need to delete and reconfigure them in the post-upgrade steps, due to UI changes in the new version.

Stop Moogfarmd

Run the following command to stop the default Moogfarmd service on any servers with the moogsoft-server package installed and where the Moogfarmd service is running. Change the service name if you are not using the default.

Run this command as root:

```
service moogfarmd stop
```



Ensure no more Moogfarmd processes are running with this command (this will force-kill any remaining java processes that have failed to stop cleanly):

```
kill -9 $(ps -ef | grep java | grep farm | awk '{print $2}') 2>/dev/null
```

Stop Apache Tomcat

Run the following command to stop the default Tomcat service on any servers with the **moogsoft-ui** package installed and where the Apache Tomcat service is running. Change the service name if you are not using the default.

Run this command as root:

```
service apache-tomcat stop
```

Stop LAMs and Integrations

Run the following command to query/stop the default LAM/integrations services on any servers with the **moogsoft-integrations/moogsoft-integrations-ui** package installed and where the LAMs/integrations are running.

Run these commands as root.

Check for running LAM and integration processes and stop them using the relevant service scripts:

```
systemctl status | grep lamd
```

```
service <lam_service_name> stop
```

Run the following command to stop any remaining active LAM and integration processes:

```
kill -9 $(ps -ef | grep java | grep _lam | awk '{print $2}') 2>/dev/null
```

Disable the Events Analyser

Run the following command to stop the Events Analyser process on any servers with the moogsoft-server package installed and where the Events Analyser is configured to run.

Run the following command as root to comment out the relevant lines in crontab:

```
(crontab -l | sed -e 's/^\(.*events_analyser.*\)$/#\1/' | crontab -
```

Run the following command to stop any active Events Analyser processes:

```
ps -ef | grep java | egrep 'events_analyser' | awk '{print $2}' | xargs  
kill 2>/dev/null
```

Back Up the Existing System

To back up the existing system:

1. Back up \$MOOGSOFT_HOME on all servers with a Cisco RPM package installed.



2. Take a snapshot (for VMs).
3. Back up MySQL on the server with the moogsoft-db RPM package installed.

Upgrade Cisco Crosswork Situation Manager

Choose the upgrade command that corresponds to your installation:

- Using a remote or local/offline Yum repository. Run the following command to perform the upgrade on every host where a Cisco RPM package has been installed:

```
yum -y upgrade $(rpm -qa --qf '%{NAME}\n' | grep moogsoft | sed 's/$/-7.2.0/'))
```

- Any host where the RPMs have been downloaded locally and are in the current folder. Run the following command to perform the upgrade:

```
yum -y upgrade moogsoft-*7.2.0*.rpm
```

Merge the Latest Configuration File Changes

Moogfarmd Configuration Change

There is a significant change to the configuration of Moogfarmd in this release. In clean installations of this release, the Moogfarmd configuration file now references Moolet configuration files located under `$MOOGSOFT_HOME/config/moolets/`.

However, this release is backwards-compatible with the old Moogfarmd configuration file format, so, you do not **need** to split up your Moogfarmd configuration as part of this upgrade, but it is recommended.

Regular Merge Process

Manually merge and compare `.rpmsave` versions of files with the new versions of those files. Add any new properties to the older versions of the files.

To find files that have been changed, moved, or deleted, run these commands:

```
find $MOOGSOFT_HOME -name '*.rpmsave'
find /etc/init.d/ -name '*.rpmsave'
```

An example command to see what differences are present in the `$MOOGSOFT_HOME/config/system.conf` file is shown below:

```
diff -u $MOOGSOFT_HOME/config/system.conf
$MOOGSOFT_HOME/config/system.conf.rpmsave
```

An example of the process to merge the differences is shown below:

1. Rename the new versions of the files, without the `.rpmsave` extension, to end with `.bak`.
2. Merge the `.rpmsave` file with the new `.bak` file by adding new properties/config where needed so the structure matches the new version of the file.



3. Rename the `.rpmsave` file to delete the `.rpmsave` extension.

Update JVM to use Java 11

On each server with a Cisco RPM package installed, you must run the following command to replace the `/usr/java/latest` symlink so it points at the new JDK11 `JAVA_HOME` directory:

```
source $MOOGSOFT_HOME/bin/utils/moog_init_functions.sh
```

If there are non-AIOps packages on this server which do not support JDK11, those applications must be updated to use a different `JAVA_HOME` symlink (not `/usr/java/latest`).

Remove References to the old MySQL Connector

The MySQL connector has been upgraded in this release.

The original connector may have been used by the 'External Database' module in the current deployment (configured in `$MOOGSOFT_HOME/config/moog_external_db_details.conf`). If this is configured on the current deployment, update it to reference the new 'mariadb' connector here: `$MOOGSOFT_HOME/lib/cots/mariadb-java-client-2.4.0.jar`

Decrease Elasticsearch Log Verbosity (Optional)

To decrease the log verbosity of Elasticsearch:

1. Run the following command to change the default log level for Elasticsearch. By default it is set to `info` but you can change it to `warn` to reduce the size of the Elasticsearch logs :

```
sed -i -e "s;rootLogger.level = info;rootLogger.level = warn;g"  
/etc/elasticsearch/log4j2.properties
```

2. Restart the Elasticsearch service:

```
service elasticsearch restart
```

Upgrade the Cisco Crosswork Situation Manager Database Schema

To upgrade the Cisco Crosswork Situation Manager database, you need to provide the Auto Upgrader utility with the credentials of a database user with super privileges. For single-host installs where MySQL was installed as part of the Cisco Crosswork Situation Manager deployment, you can use the default 'root' user.

1. Run the following command (on the server where the `moogsoft-db` package is installed) after substituting the `<MySQL-SuperUsername>` argument:

```
bash $MOOGSOFT_HOME/bin/utils/moog_db_auto_upgrader -t 7.2.0 -u <MySQL-SuperUsername>
```



2. Enter the password for that user. You can provide the password to the utility with the `-p` flag but this is not recommended in non-test deployments for security reasons.

Drop Deprecated Historic Database Tables

Run the following commands to drop two tables from the historic database that are no longer used. If they are not dropped at this stage, the `moog_db_validator` utility at the end of this process will report their presence as a delta.

Run these commands on the server with the `moogsoft-db` package installed:

```
bash $MOOGSOFT_HOME/bin/utils/moog_mysql_client -i -e "drop table room_post_sigs";
bash $MOOGSOFT_HOME/bin/utils/moog_mysql_client -i -e "drop table room_posts";
```

Update the RabbitMQ Configuration

Update the RabbitMQ configuration to support autoheal and other processes that improve stability in HA environments. To do this, run the following commands on each server the `moogsoft-mooms` / RabbitMQ package is installed on. Replace `<VHOST>` with the name of your RabbitMQ vhost:

```
cp -f $MOOGSOFT_HOME/etc/cots/rabbitmq/rabbitmq.config /etc/rabbitmq/
bash $MOOGSOFT_HOME/bin/utils/moog_init_mooms.sh -z <VHOST> -p
```

Enable the Graph Analyser Crontab (Optional)

If you are using a topology with Cisco Crosswork Situation Manager, it is important to keep the vertex entropy values up to date as the topology changes. This can be done with a cronjob.

Run the following command on the server where the `moogsoft-server` package is deployed (where `Moogfarmd` is running):

```
$MOOGSOFT_HOME/bin/utils/moog_init_server.sh -g
```

Upgrade Apache Tomcat and the Webapps

Cisco Crosswork Situation Manager v7.2.0 ships with Apache Tomcat version 9.0.13.0.

Run the commands in this section on the server with the `moogsoft-ui` RPM package installed on it.

1. Remove the existing Apache Tomcat:

```
rm -rf /etc/init.d/apache-tomcat;
rm -rf $APPSERVER_HOME
rm -rf /usr/share/apache-tomcat
```

2. Deploy the new version of Apache Tomcat:



```
$MOOGSOFT_HOME/bin/utils/moog_init_ui.sh -tfw
```

If the webapps do not extract and the connection to MySQL is SSL-enabled, it is likely that the hostname of the self-signed certificates does not match the ones set. See [Encrypt Database Communications](#) for steps to resolve the issue.

3. Restart Nginx:

```
service nginx restart
```

Update and Reconfigure Integrations

This upgrade process replaces the Moobot file

`$MOOGSOFT_HOME/bots/moobots/RemedyIntegration.js`. If you have customised this file, back it up before continuing.

Run the commands in this section on the server with the **moogsoft-integrations-ui** package installed on it.

To update the integrations:

1. The LAMs/Integrations no longer use the 'state' files under `$MOOGSOFT_HOME/config`. Remove these files by running the following commands:

```
rm -f $MOOGSOFT_HOME/config/*_lam.state;
rm -f $MOOGSOFT_HOME/config/{null.state, TestStateManager.state}
```

2. Run the following command to extract the old and new Integrations:

```
bash $MOOGSOFT_HOME/bin/utils/integration_installer -a -l WARN;
```

In the next step you will reconfigure the ServiceNow integration which will restart the MID server.

Configure the ServiceNow MID Server to use Java 8

If you are using a ServiceNow MID server installed on the same host as Cisco Crosswork Situation Manager, you must configure it to point to Java 8. The MID server requires Java 8 (update 152 or later). It will not work with Java 9+. To do this:

1. Install the latest version of Java 8. See the [ServiceNow MID server system requirements](#) for more information.
2. Stop the MID server by running the appropriate command. For example:

```
kill -9 $(ps -ef | grep mid_server | grep -v grep | awk '{print $2}')
```

3. Configure the `wrapper.java.command` property to point to the Java 8 binary in the following file:

```
/usr/local/servicenow/moog_mid_server/agent/conf/wrapper-override.conf.
```

For example:



```
wrapper.java.command=/usr/java/jre1.8.0_171-amd64/bin/java
```

Reconfigure your UI Integrations

Delete and reconfigure all of your UI integrations, using the configuration details you noted prior to the upgrade. This is required due to UI changes in the new version.

Confirm All Services are Running and Start Moogfarmd

To confirm all services are running:

1. Run the following commands to check all required services are running and start Moogfarmd:

```
service apache-tomcat status
service moogfarmd start
service nginx status
service elasticsearch status
```

2. Run the following command to re-enable the Events Analyser cronjobs:

```
(crontab -l | sed -e 's/^#\+\.(*events_analyser.*\)/\1/') | crontab -
```

3. Run the following command to restart any previously running Integrations:

```
service moogstartupd restart
```

Add the Keepalive Cronjob (Optional)

A new cronjob is installed for clean installs of 7.2.0 which provides the ability for certain services to be restarted automatically should they fail. For RPM deployments this is usually handled by chkconfig or MySQL itself for example, but can be useful in certain cases. See [Configure Services to Restart](#) for more information.

Command to enable if required:

```
(crontab -l; echo -e "*\t*\t*\t*\t*\n$MOOGSOFT_HOME/bin/utils/process keepalive.sh 2>&1") | crontab -
```

Verify the Upgrade

You can verify the upgrade manually or using automatic utilities.

Verify the Upgrade Manually

Perform the following basic steps to ensure that the upgrade to Cisco Crosswork Situation Manager v7.2.0 was successful:

1. Check that the UI login page displays "Version 7.2.0" at the top.
2. Check that the UI "Support Information" window correctly indicates the current version as "7.2.0" and shows the correct schema upgrade history.



Verify the Upgrade Using Automatic Utilities

Run the following automatic utilities to ensure that the upgrade to Cisco Crosswork Situation Manager v7.2.0 was successful:

1. Confirm that all Cisco Crosswork Situation Manager files have been deployed correctly within \$MOOGSOFT_HOME using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_install_validator.sh
```

2. Confirm that all Apache Tomcat files have been deployed correctly within \$MOOGSOFT_HOME using this utility:

```
$MOOGSOFT_HOME/bin/utils/tomcat_install_validator.sh
```

- If there are some web app differences, you can resolve them using `moog_init_ui.sh -w` which extracts the web apps with the right files.

3. Confirm that the database schema has been upgraded successfully using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_db_validator.sh
```

If errors such as the following are seen:

Differences found in 'moogdb' tables:

57a58

> key 'filter_id' ('filter_id'),

194a196

> key 'enrichment_static_mappings_ibfk_1' ('eid'),

1196a1199

> key 'sig_id' ('sig_id'),

1325a1329

> key 'filter_id' ('filter_id'),

Then run the following commands to fix the index issues:

```
mysql -u root -e "use moogdb; alter table alert_filters_access drop key \ 'filter_id\ '"
```

```
mysql -u root -e "use moogdb; alter table situation_filters_access drop key \ 'filter_id\ '"
```

```
mysql -u root -e "use moogdb; alter table enrichment_static_mappings drop key \ 'enrichment_static_mappings_ibfk_1\ '"
```

```
mysql -u root -e "use moogdb; alter table sig_stats_cache drop key \ 'sig_id\ '"
```

4. Confirm that all the steps have completed successfully.

Some schema differences may be valid (e.g. custom_info related). If there are more substantial differences, you should investigate further to verify that all the prerequisite upgrade scripts have been applied in the right order:

```
mysql -u root <moogdb_database_name> -e "select * from schema_upgrades;"
```



```
mysql -u root <moog_reference_database_name> -e "select * from  
schema_upgrades;"
```

Remove the MySQL Connector

After upgrading to CCSM 7.2 go to all your machines and remove the file:

```
$MOOGSOFT_HOME/lib/cots/nonDist/mysql-connector-java-  
5.1.45.jar
```

Troubleshooting

If you have any issues, refer to [Troubleshooting](#).

Upgrade RPM v7.0.x to RPM v7.2.x

This topic describes the upgrade procedure of RPM deployments from Cisco Crosswork Situation Manager v7.0.x to Cisco Crosswork Situation Manager v7.2.0. Refer to [Upgrade - RPM Deployments](#) for general information and links to upgrades for other versions.

Before You Begin

Before you begin the upgrade process, complete the following prerequisite steps as appropriate.

Prepare for an Offline Upgrade

To prepare for an offline upgrade, where the Cisco Crosswork Situation Manager packages reside in a 'local' Yum repository, follow steps 1-8 in section "Prepare the Local yum Repositories" in the [Pre-Install Moogsoft AIOps - Offline RPMs](#) guide. Then follow this guide to perform the upgrade. Take into account any step-specific notes regarding offline upgrades.

Note UI Integrations Configuration

If you are using any UI integrations, make a note of their connection and configuration details before you begin the upgrade. You will need to delete and reconfigure them in the post-upgrade steps, due to UI changes in the new version.

Stop Moogfarmd

Run the following command to stop the default Moogfarmd service on any servers with the moogsoft-server package installed and where the Moogfarmd service is running. Change the service name if you are not using the default.

Run this command as root:

```
service moogfarmd stop
```

Ensure no more Moogfarmd processes are running with this command (this will force-kill any remaining java processes that have failed to stop cleanly):

```
kill -9 $(ps -ef | grep java | grep farm | awk '{print $2}') 2>/dev/null
```



Stop Apache Tomcat

Run the following command to stop the default Tomcat service on any servers with the **moogsoft-ui** package installed and where the Apache Tomcat service is running. Change the service name if you are not using the default.

Run this command as root:

```
service apache-tomcat stop
```

Stop LAMs and Integrations

Run the following command to query/stop the default LAM/integrations services on any servers with the **moogsoft-integrations/moogsoft-integrations-ui** package installed and where the LAMs/integrations are running.

Run these commands as root.

Check for running LAM and integration processes and stop them using the relevant service scripts:

```
systemctl status | grep lamd
```

```
service <lam_service_name> stop
```

Ensure no more Moogfarmd processes are running with this command (this will force-kill any remaining java processes that have failed to stop cleanly):

```
kill -9 $(ps -ef | grep java | grep _lam | awk '{print $2}') 2>/dev/null
```

Disable the Events Analyser

Run the following command to stop the Events Analyser process on any servers with the **moogsoft-server** package installed and where the Events Analyser is configured to run.

Run the following command as root to comment out the relevant lines in crontab:

```
(crontab -l | sed -e 's/^\(.*events_analyser.*\)$/#\1/' ) | crontab -
```

Run the following command to stop any active Events Analyser processes:

```
ps -ef | grep java | egrep 'events_analyser' | awk '{print $2}' | xargs  
kill 2>/dev/null
```

Back Up the Existing System

To back up the existing system:

1. Back up \$MOOGSOFT_HOME on all servers with a moogsoft RPM package installed.
2. Take a snapshot (for VMs).
3. Back up MySQL on the server with the moogsoft-db RPM package installed



Upgrade Cisco Crosswork Situation Manager

Choose the upgrade command that corresponds to your installation:

- Using a remote or local/offline Yum repository. Run the following command to perform the upgrade on every host where a Cisco RPM package has been installed:

```
yum -y upgrade $(rpm -qa --qf '%{NAME}\n' | grep moogsoft | sed 's/$/-7.2.0/')
```

- Any host where the RPMs have been downloaded locally and are in the current folder. Run the following command to perform the upgrade:

```
yum -y upgrade moogsoft-*7.2.0*.rpm
```

Merge the Latest Configuration File Changes

Moogfarmd Configuration Change

There is a significant change to the configuration of Moogfarmd in this release. In clean installations of this release, the Moogfarmd configuration file now references Moolet configuration files located under `$MOOGSOFT_HOME/config/moolets/`.

However, this release is backwards-compatible with the old Moogfarmd configuration file format, so, you do not **need** to split up your Moogfarmd configuration as part of this upgrade, but it is recommended.

Regular Merge Process

Manually merge and compare `.rpmsave` versions of files with the new versions of those files. Add any new properties to the older versions of the files.

To find files that have been changed, moved, or deleted, run these commands:

```
find $MOOGSOFT_HOME -name '*.rpmsave'
find /etc/init.d/ -name '*.rpmsave'
```

An example command to see what differences are present in the `$MOOGSOFT_HOME/config/system.conf` file is shown below:

```
diff -u $MOOGSOFT_HOME/config/system.conf
$MOOGSOFT_HOME/config/system.conf.rpmsave
```

An example of the process to merge the differences is shown below:

1. Rename the new versions of the files, without the `.rpmsave` extension, to end with `.bak`.
2. Merge the `.rpmsave` file with the new `.bak` file by adding new properties/config where needed so the structure matches the new version of the file.
3. Rename the `.rpmsave` file to delete the `.rpmsave` extension.



Update JVM to use Java 11

On each server with a moogsoft RPM package installed, you must run the following command to replace the `/usr/java/latest` symlink so it points at the new JDK11 `JAVA_HOME` directory:

```
source $MOOGSOFT_HOME/bin/utils/moog_init_functions.sh
```

If there are non-AIOps packages on this server which do not support JDK11, those applications must be updated to use a different `JAVA_HOME` symlink (not `/usr/java/latest`).

Remove References to the old MySQL Connector

The MySQL connector has been upgraded in this release.

The original connector might have been used by the 'External Database' module in the current deployment (configured in `$MOOGSOFT_HOME/config/moog_external_db_details.conf`). If this is configured on the current deployment, it should be updated to reference the new 'mariadb' connector here: `$MOOGSOFT_HOME/lib/cots/mariadb-java-client-2.4.0.jar`

Upgrade the Cisco Crosswork Situation Manager Database Schema

To upgrade the Cisco Crosswork Situation Manager database, you need to provide the Auto Upgrader utility with the credentials of a database user with super privileges. For single-host installs where MySQL was installed as part of the Cisco Crosswork Situation Manager deployment, you can use the default 'root' user.

1. Run the following command (on the server where the moogsoft-db package is installed) after substituting the `<MySQL-SuperUsername>` argument:

```
bash $MOOGSOFT_HOME/bin/utils/moog_db_auto_upgrader -t 7.2.0 -u <MySQL-SuperUsername>
```

2. Enter the password for that user. You can provide the password to the utility with the `-p` flag but this is not recommended in non-test deployments for security reasons.

Drop Deprecated Historic Database Tables

Run the following commands to drop two tables from the historic database that are no longer used. If they are not dropped at this stage, the `moog_db_validator` utility at the end of this process will report their presence as a delta.

Run these commands on the server with the moogsoft-db package installed:

```
bash $MOOGSOFT_HOME/bin/utils/moog_mysql_client -i -e "drop table room_post_sigs";
bash $MOOGSOFT_HOME/bin/utils/moog_mysql_client -i -e "drop table room_posts";
```



Update the RabbitMQ Configuration

Update the RabbitMQ configuration to support autoheal and other processes that improve stability in HA environments. To do this, run the following commands on each server the moogsoft-mooms / RabbitMQ package is installed on. Replace <VHOST> with the name of your RabbitMQ vhost:

```
cp -f $MOOGSOFT_HOME/etc/cots/rabbitmq/rabbitmq.config /etc/rabbitmq/  
bash $MOOGSOFT_HOME/bin/utils/moog_init_mooms.sh -z <VHOST> -p
```

Enable the Graph Analyser Crontab (Optional)

If you are using a topology with Cisco Crosswork Situation Manager, it is important to keep the vertex entropy values up to date as the topology changes. This can be done with a cronjob.

Run the following command on the server where the moogsoft-server package is deployed (where Moogfarmd is running):

```
$MOOGSOFT_HOME/bin/utils/moog_init_server.sh -g
```

Upgrade Apache Tomcat and the Webapps

Cisco Crosswork Situation Manager v7.2.0 ships with Apache Tomcat version 9.0.13.0.

Run the commands in this section on the server with the moogsoft-ui RPM package installed on it.

1. Remove the existing Apache Tomcat:

```
rm -rf /etc/init.d/apache-tomcat;  
rm -rf $APPSERVER_HOME  
rm -rf /usr/share/apache-tomcat
```

2. Deploy the new version of Apache Tomcat:

```
$MOOGSOFT_HOME/bin/utils/moog_init_ui.sh -tfw
```

If the webapps do not extract and the connection to MySQL is SSL-enabled, it is likely that the hostname of the self-signed certificates does not match the ones set. See [Encrypt Database Communications](#) for steps to resolve the issue.

3. Restart Nginx:

```
service nginx restart
```

Update and Reconfigure Integrations

This upgrade process replaces the Moobot file

`$MOOGSOFT_HOME/bots/moobots/RemedyIntegration.js`. If you have customised this file, back it up before continuing.



Run the commands in this section on the server with the **moogsoft-integrations-ui** package installed on it.

To update the integrations:

1. The LAMs/Integrations no longer use the 'state' files under `$MOOGSOFT_HOME/config`. Remove these files by running the following commands:

```
rm -f $MOOGSOFT_HOME/config/*_lam.state;  
rm -f $MOOGSOFT_HOME/config/{null.state, TestStateManager.state}
```

2. Run the following command to extract the old and new Integrations:

```
bash $MOOGSOFT_HOME/bin/utils/integration_installer -a -l WARN;
```

Configure the ServiceNow MID Server to use Java 8

If you are using a ServiceNow MID server installed on the same host as Cisco Crosswork Situation Manager, you must configure it to point to Java 8. The MID server requires Java 8 (update 152 or later). It will not work with Java 9+. To do this:

1. Install the latest version of Java 8. See the [ServiceNow MID server system requirements](#) for more information.

2. Stop the MID server by running the appropriate command. For example:

```
kill -9 $(ps -ef | grep mid_server | grep -v grep | awk '{print $2}')
```

3. Configure the `wrapper.java.command` property to point to the Java 8 binary in the following file:

```
/usr/local/servicenow/moog_mid_server/agent/conf/wrapper-override.conf.
```

For example:

```
wrapper.java.command=/usr/java/jre1.8.0_171-amd64/bin/java
```

In the next step you will reconfigure the ServiceNow integration which will restart the MID server.

Reconfigure your UI Integrations

Delete and reconfigure all of your UI integrations, using the configuration details you noted prior to the upgrade. This is required due to UI changes in the new version.

Confirm All Services are Running and Start Moogfarmd

To confirm all services are running:

1. Run the following commands to check all required services are running and start Moogfarmd:



```
service apache-tomcat status
service moogfarmd start
service nginx status
service elasticsearch status
```

2. Run the following command to re-enable the Events Analyser cronjobs:

```
(crontab -l | sed -e 's/^\#\+\(.*events_analyser.*\)/\1/' | crontab -
```

3. Run the following command to restart any previously running Integrations:

```
service moogstartupd restart
```

Add the Keepalive Cronjob (Optional)

A new cronjob is installed for clean installs of 7.2.0 which provides the ability for certain services to be restarted automatically should they fail. For RPM deployments this is usually handled by chkconfig or mysql itself for example, but can be useful in certain cases. See [Configure Services to Restart](#) for more information.

Command to enable if required:

```
(crontab -l; echo -e "*\t*\t*\t*\t*
$MOOGSOFT_HOME/bin/utils/process_keepalive.sh 2>&1") | crontab -
```

Verify the Upgrade

You can verify the upgrade manually or using automatic utilities.

Verify the Upgrade Manually

Perform the following basic steps to ensure that the upgrade to Cisco Crosswork Situation Manager v7.2.0 was successful:

1. Check that the UI login page displays "Version 7.2.0" at the top.
2. Check that the UI "Support Information" window correctly indicates the current version as "7.2.0" and shows the correct schema upgrade history.

Verify the Upgrade Using Automatic Utilities

Run the following automatic utilities to ensure that the upgrade to Cisco Crosswork Situation Manager v7.2.0 was successful:

1. Confirm that all Cisco Crosswork Situation Manager files have been deployed correctly within \$MOOGSOFT_HOME using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_install_validator.sh
```

2. Confirm that all Apache Tomcat files have been deployed correctly within \$MOOGSOFT_HOME using this utility:

```
$MOOGSOFT_HOME/bin/utils/tomcat_install_validator.sh
```



- If there are some web app differences, you can resolve them using `moog_init_ui.sh -w` which extracts the web apps with the right files.

3. Confirm that the database schema has been upgraded successfully using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_db_validator.sh
```

If errors such as the following are seen:

Differences found in 'moogdb' tables:

57a58

```
> key 'filter_id' ('filter_id'),
```

194a196

```
> key 'enrichment_static_mappings_ibfk_1' ('eid'),
```

1196a1199

```
> key 'sig_id' ('sig_id'),
```

1325a1329

```
> key 'filter_id' ('filter_id'),
```

Then run the following commands to fix the index issues:

```
mysql -u root -e "use moogdb; alter table alert_filters_access drop key \ 'filter_id\ '"
```

```
mysql -u root -e "use moogdb; alter table situation_filters_access drop key \ 'filter_id\ '"
```

```
mysql -u root -e "use moogdb; alter table enrichment_static_mappings drop key \ 'enrichment_static_mappings_ibfk_1\ '"
```

```
mysql -u root -e "use moogdb; alter table sig_stats_cache drop key \ 'sig_id\ '"
```

4. Confirm that all the steps have completed successfully.

Some schema differences may be valid (e.g. `custom_info` related). If there are more substantial differences, you should investigate further to verify that all the prerequisite upgrade scripts have been applied in the right order:

```
mysql -u root <moogdb_database_name> -e "select * from schema_upgrades;"
```

```
mysql -u root <moog_reference_database_name> -e "select * from schema_upgrades;"
```

Remove the MySQL Connector

After upgrading to CCSM 7.2 go to all your machines and remove the file:

```
$MOOGSOFT_HOME/lib/cots/nonDist/mysql-connector-java-5.1.45.jar
```

Troubleshooting

If you have any issues, refer to [Troubleshooting](#).



Upgrade RPM v7.1.x to RPM v7.2.x

This topic describes the upgrade procedure of RPM deployments from Cisco Crosswork Situation Manager v7.1.x to Cisco Crosswork Situation Manager v7.2.0. Refer to [Upgrade - RPM Deployments](#) for general information and links to upgrades for other versions.

Before You Begin

Before you begin the upgrade process, complete the following prerequisite steps as appropriate.

[Prepare for an Offline Upgrade](#)

To prepare for an offline upgrade, where the Cisco Crosswork Situation Manager packages reside in a 'local' Yum repository, follow steps 1-8 in section "Prepare the Local yum Repositories" in the [Pre-Install Moogsoft AIOps - Offline RPMs](#) guide. Then follow this guide to perform the upgrade. Take into account any step-specific notes regarding offline upgrades.

[Note UI Integrations Configuration](#)

If you are using any of the following UI integrations, make a note of their connection and configuration details before you begin the upgrade. You will need to delete and reconfigure them in the post-upgrade steps, due to UI changes in the new version.

- AWS CloudWatch
- Cherwell
- JIRA Service Desk
- JIRA Software
- JMS
- New Relic
- Remedy
- ServiceNow
- SevOne
- Slack
- SolarWinds
- VMware vCenter
- VMware vSphere
- vRealize Log Insight



- WebSphere MQ
- xMatters

Stop Moogfarmd

Run the following command to stop the default Moogfarmd service on any servers with the **moogsoft-server** package installed and where the Moogfarmd service is running. Change the service name if you are not using the default.

Run this command as root:

```
service moogfarmd stop
```

Ensure no more Moogfarmd processes are running with this command (this will force-kill any remaining java processes which have failed to stop cleanly):

```
kill -9 $(ps -ef | grep java | grep farm | awk '{print $2}') 2>/dev/null
```

Stop Apache Tomcat

Run the following command to stop the default tomcat service on any servers with the **moogsoft-ui** package installed and where the Apache Tomcat service is running. Change the service name if you are not using the default.

Run this command as root:

```
service apache-tomcat stop
```

Stop LAMs and Integrations

Run the following command to query/stop the default LAM/integrations services on any servers with the **moogsoft-integrations/moogsoft-integrations-ui** package installed and where the LAMs/integrations are running.

Run these commands as root.

Check for running LAM and integration processes and stop them using the relevant service scripts:

```
systemctl status | grep lamd
```

```
service <lam_service_name> stop
```

Run the following command to stop any remaining active LAM and integration processes:

```
kill -9 $(ps -ef | grep java | grep _lam | awk '{print $2}') 2>/dev/null
```

Disable the Events Analyser

Run the following command to stop the Events Analyser process on any servers with the **moogsoft-server** package installed and where the Events Analyser is configured to run.



Run the following command as root to comment out the relevant lines in crontab:

```
(crontab -l | sed -e 's/^(\.*events_analyser.*\)$/#\1/' ) | crontab -
```

Run the following command to stop any active Events Analyser processes:

```
ps -ef | grep java | egrep 'events_analyser' | awk '{print $2}' | xargs  
kill 2>/dev/null
```

Back Up the Existing System

To back up the existing system:

1. Back up \$MOOGSOFT_HOME on all servers with a moogsoft RPM package installed.
2. Take a snapshot (for VMs).
3. Back up MySQL on the server with the moogsoft-db RPM package installed.

Upgrade Cisco Crosswork Situation Manager

Choose the upgrade command that corresponds to your preferred upgrade mechanism:

- Using a remote or local/offline Yum repository. Run the following command to perform the upgrade on every host where a moogsoft RPM package has been installed:

```
yum -y upgrade $(rpm -qa --qf '%{NAME}\n' | grep moogsoft | sed 's/$/-  
7.2.0/')
```

- Any host where the RPMs have been downloaded locally and are in the current folder. Run the following command to perform the upgrade:

```
yum -y upgrade moogsoft-*7.2.0*.rpm
```

Merge the Latest Configuration File Changes

Manually merge and compare .rpmsave versions of files with the new versions of those files. Add any new properties to the older versions of the files.

To find files that have been changed, moved, or deleted, run these commands:

```
find $MOOGSOFT_HOME -name '*.rpmsave'  
find /etc/init.d/ -name '*.rpmsave'
```

An example command to see what differences are present in the \$MOOGSOFT_HOME/config/system.conf file is shown below:

```
diff -u $MOOGSOFT_HOME/config/system.conf  
$MOOGSOFT_HOME/config/system.conf.rpmsave
```

An example of the process to merge the differences is shown below:

1. Rename the new versions of the files, without the .rpmsave extension, to end with .bak.



2. Merge the `.rpmsave` file with the new `.bak` file by adding new properties/config where needed so the structure matches the new version of the file.
3. Rename the `.rpmsave` file to delete the `.rpmsave` extension.

Update JVM to use Java 11

On each server with a moogsoft RPM package installed, you must run the following command to replace the `/usr/java/latest` symlink so it points at the new JDK11 `JAVA_HOME` directory:

```
source $MOOGSOFT_HOME/bin/utils/moog_init_functions.sh
```

If there are non-AIOps packages on this server which do not support JDK11, those applications must be updated to use a different `JAVA_HOME` symlink (not `/usr/java/latest`).

Remove References to the old MySQL Connector

The MySQL connector has been upgraded in this release.

The original connector may have been used by the 'External Database' module in the current deployment (configured in `$MOOGSOFT_HOME/config/moog_external_db_details.conf`). If this is configured on the current deployment, update it to reference the new 'mariadb' connector here: `$MOOGSOFT_HOME/lib/cots/mariadb-java-client-2.4.0.jar`

Upgrade the Cisco Crosswork Situation Manager Database Schema

To upgrade the Cisco Crosswork Situation Manager database, you need to provide the Auto Upgrader utility with the credentials of a database user with super privileges. For single-host installs where MySQL was installed as part of the Cisco Crosswork Situation Manager deployment, you can use the default 'root' user.

1. Run the following command (on the server where the moogsoft-db package is installed) after substituting the `<MySQL-SuperUsername>` argument:

```
bash $MOOGSOFT_HOME/bin/utils/moog_db_auto_upgrader -t 7.2.0 -u <MySQL-SuperUsername>
```

2. Enter the password for that user. You can provide the password to the utility with the `-p` flag but this is not recommended in non-test deployments for security reasons.

Enable the Graph Analyser Crontab (Optional)

If you are using a topology with Cisco Crosswork Situation Manager, it is important to keep the vertex entropy values up to date as the topology changes. This can be done with a cronjob.

Run the following command on the server where the moogsoft-server package is deployed (where Moogfarmd is running):



```
$MOOGSOFT_HOME/bin/utils/moog_init_server.sh -g
```

Upgrade Apache Tomcat and the Webapps

Cisco Crosswork Situation Manager v7.2.0 ships with Apache Tomcat version 9.0.13.0.

Run the commands in this section on the server with the moogsoft-ui RPM package installed on it.

1. Remove the existing Apache Tomcat:

```
rm -rf /etc/init.d/apache-tomcat;  
rm -rf $APPSERVER_HOME  
rm -rf /usr/share/apache-tomcat
```

2. Deploy the new version of Apache Tomcat:

```
$MOOGSOFT_HOME/bin/utils/moog_init_ui.sh -tfw
```

If the webapps do not extract and the connection to MySQL is SSL-enabled, it is likely that the hostname of the self-signed certificates does not match the ones set. See [Encrypt Database Communications](#) for steps to resolve the issue.

3. Restart Nginx:

```
service nginx restart
```

Update and Reconfigure Integrations

This upgrade process replaces the Moobot file

`$MOOGSOFT_HOME/bots/moobots/RemedyIntegration.js`. If you have customised this file, back it up before continuing.

Run the commands in this section on the server with the **moogsoft-integrations-ui** package installed on it.

To update the integrations:

1. The LAMs/Integrations no longer use the 'state' files under `$MOOGSOFT_HOME/config`. Remove these files by running the following commands:

```
rm -f $MOOGSOFT_HOME/config/*_lam.state;  
rm -f $MOOGSOFT_HOME/config/{null.state, TestStateManager.state}
```

2. Run the following command to extract the old and new Integrations:

```
bash $MOOGSOFT_HOME/bin/utils/integration_installer -a -l WARN;
```

Configure the ServiceNow MID Server to use Java 8

If you are using a ServiceNow MID server installed on the same host as Cisco Crosswork Situation Manager, you must configure it to point to Java 8. The MID server requires Java 8 (update 152 or later). It will not work with Java 9+. To do this:



1. Install the latest version of Java 8. See the [ServiceNow MID server system requirements](#) for more information.
2. Stop the MID server by running the appropriate command. For example:

```
kill -9 $(ps -ef | grep mid_server | grep -v grep | awk '{print $2}')
```
3. Configure the `wrapper.java.command` property to point to the Java 8 binary in the following file:

```
/usr/local/servicenow/moog_mid_server/agent/conf/wrapper-override.conf.
```

For example:

```
wrapper.java.command=/usr/java/jre1.8.0_171-amd64/bin/java
```

In the next step you will reconfigure the ServiceNow integration which will restart the MID server.

[Reconfigure your UI Integrations](#)

If you are using any of the following UI integrations, delete and reconfigure them using the details you noted prior to the upgrade. This is required due to UI changes in the new version.

- AWS CloudWatch
- Cherwell
- JIRA Service Desk
- JIRA Software
- JMS
- New Relic
- Remedy
- ServiceNow
- SevOne
- Slack
- SolarWinds
- VMware vCenter
- VMware vSphere
- vRealize Log Insight
- WebSphere MQ



- xMatters

Confirm All Services are Running and Start Moogfarmd

To confirm all services are running:

1. Run the following commands to check all required services are running and start Moogfarmd:

```
service apache-tomcat status
service moogfarmd start
service nginx status
service elasticsearch status
```

2. Run the following command to re-enable the events_analyser cronjobs:

```
(crontab -l | sed -e 's/^#\+\.events_analyser.*\)/\1/' | crontab -
```

3. Run the following command to restart any previously running Integrations:

```
service moogstartupd restart
```

Verify the Upgrade

You can verify the upgrade manually or using automatic utilities.

Verify the Upgrade Manually

Perform the following basic steps to ensure that the upgrade to Cisco Crosswork Situation Manager v7.2.0 was successful:

1. Check that the UI login page displays "Version 7.2.0" at the top.
2. Check that the UI "Support Information" window correctly indicates the current version as "7.2.0" and shows the correct schema upgrade history.

Verify the Upgrade Using Automatic Utilities

Run the following automatic utilities to ensure that the upgrade to Cisco Crosswork Situation Manager v7.2.0 was successful:

1. Confirm that all Cisco Crosswork Situation Manager files have been deployed correctly within \$MOOGSOFT_HOME using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_install_validator.sh
```

2. Confirm that all Apache Tomcat files have been deployed correctly within \$MOOGSOFT_HOME using this utility:

```
$MOOGSOFT_HOME/bin/utils/tomcat_install_validator.sh
```

- If there are some web app differences, you can resolve them using `moog_init_ui.sh -w` which extracts the web apps with the right files.



3. Confirm that the database schema has been upgraded successfully using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_db_validator.sh
```

If errors such as the following are seen:

Differences found in 'moogdb' tables:

```
57a58
```

```
> key 'filter_id' ('filter_id'),
```

```
194a196
```

```
> key 'enrichment_static_mappings_ibfk_1' ('eid'),
```

```
1196a1199
```

```
> key 'sig_id' ('sig_id'),
```

```
1325a1329
```

```
> key 'filter_id' ('filter_id'),
```

Then run the following commands to fix the index issues:

```
mysql -u root -e "use moogdb; alter table alert_filters_access drop key \
```

```
mysql -u root -e "use moogdb; alter table situation_filters_access drop key \
```

```
mysql -u root -e "use moogdb; alter table enrichment_static_mappings drop key \
```

```
mysql -u root -e "use moogdb; alter table sig_stats_cache drop key \
```

4. Confirm that all the steps have completed successfully.

Some schema differences may be valid (e.g. custom_info related). If there are more substantial differences, you should investigate further to verify that all the pre-requisite upgrade scripts have been applied in the right order:

```
mysql -u root <moogdb_database_name> -e "select * from schema_upgrades;"
```

```
mysql -u root <moog_reference_database_name> -e "select * from schema_upgrades;"
```

Remove the MySQL Connector

After upgrading to CCSM 7.2 go to all your machines and remove the file:

```
$MOOGSOFT_HOME/lib/cots/nonDist/mysql-connector-java-5.1.45.jar
```

Troubleshooting

If you have any issues, refer to [Troubleshooting](#).

High Availability Overview

Cisco Crosswork Situation Manager supports high availability (HA) architectures to make the system more fault tolerant. Each component supports a multi-node

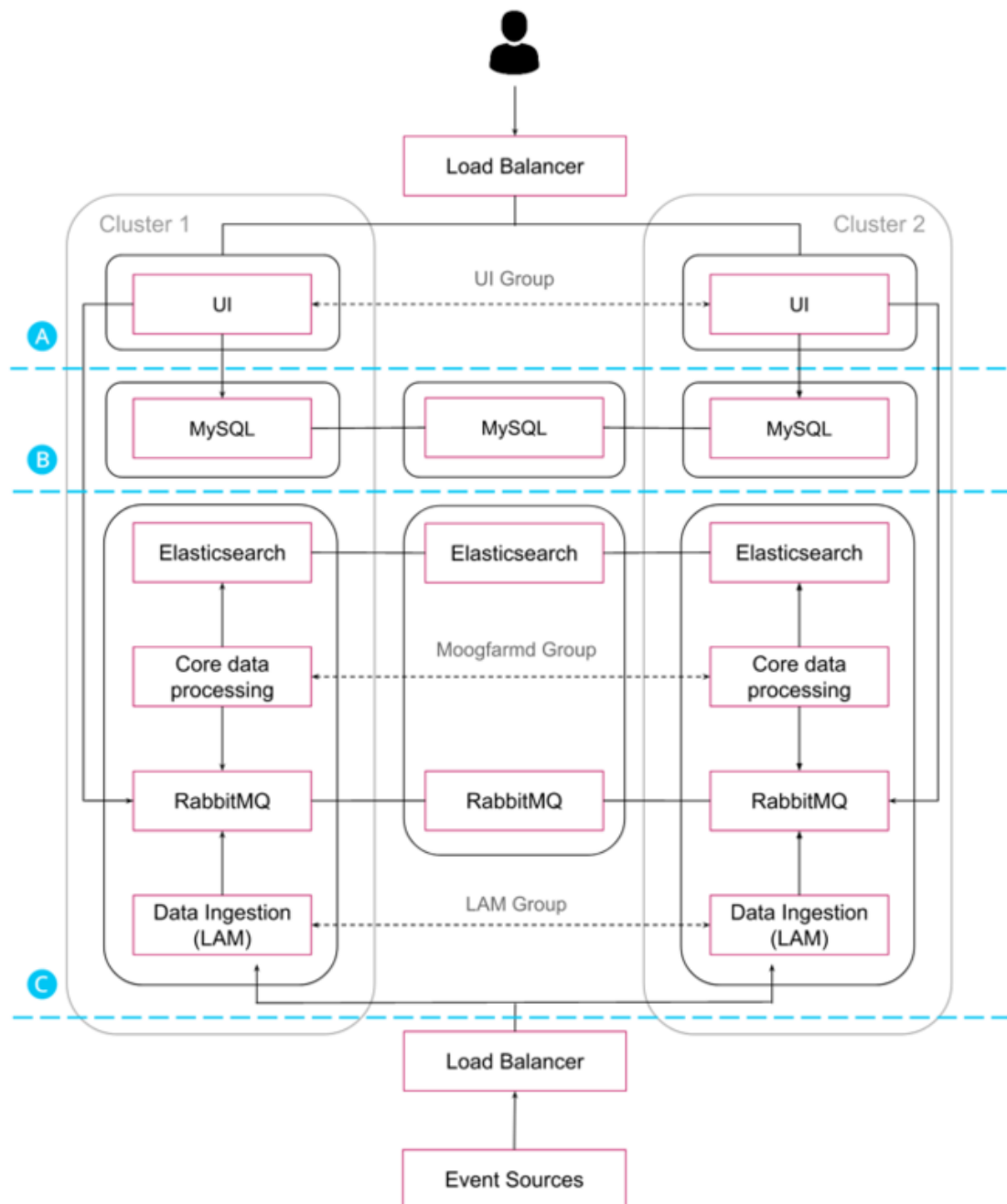


architecture to enable redundancy, failover, or both to minimize risk of data loss. For example, in the case of a hardware failure.

This topic covers the architectures you can use for to achieve HA with Cisco Crosswork Situation Manager . For an example of how to set up a single site HA system, see [Single Site High Availability](#). See [High Availability Reference Architecture](#) for detailed diagram of the components in a single site HA configuration.

HA Architecture Basics

In its simplest form, each Moogsoft AIOps cluster runs on a designated host and the two hosts constitute an HA pair. The third party components Elasticsearch, RabbitMQ, and the database require an additional node to achieve HA. The following diagram illustrates a single site scenario for HA using two full stacks of Moogsoft AIOps, a primary cluster and a secondary cluster:



Distributed HA Architectures

Moogsoft AIOps supports high availability in distributed architectures where different machines host a subset of the stack. The dotted lines in the diagram illustrate one option dividing clusters into a distributed architecture. For example you can separate the UI stack **(A)** from the database **(B)** and the database from the remaining layers **(C)**. Alternatively, you can run LAMs for data ingestion on separate hosts.

When choosing how to distribute the components of your HA deployment onto multiple hosts, you must run the following components on the same host:



- The UI stack (Nginx and Tomcat) .
- Data processing (Moogfarmd), the message bus (RabbitMQ), and search (Elasticsearch).

The database and LAMs can run on hosts with no other components. However, to minimize latency between Moogfarmd and the database, it is a good idea to run both components on the same host. If you choose to run the database separately from Moogfarmd, there should be no more than 5 ms latency between components.

To increase capacity within the HA architecture, you can:

- Add memory or CPU to hosts running Moogfarmd. If you add additional machines beyond a single HA pair, you increase redundancy but not capacity.
- Add hosts to increase LAM processing capacity. If you implement this model, do not extend the LAM capacity beyond the capacity of Moogfarmd to process incoming events and alerts.

See [Sizing Recommendations](#) for more information on hardware sizes and capacity.

After you decide on the best HA architecture for your environment, you can plan your implementation.

High Availability Configuration Hierarchy

Cisco Crosswork Situation Manager deployments use a tiered hierarchy of clusters, groups, and instances to achieve High Availability.

A **cluster** is a collection of Cisco Crosswork Situation Manager Moogsoft AIOps components. To achieve HA you need at least two clusters that include all the Cisco Crosswork Situation Manager components. You need an additional, third machine, for message queue, search, and database components.

A **group** is a set of identical components that provide resilience over two or more clusters. Cisco Crosswork Situation Manager automatically controls the active or passive behavior and failover of the instances within a group. See

</document/preview/77311#UIDaea0bdc20a2d5c12ffc6fce1c1d3b562> for more information. Modes of Operation in High Availability

An example of a group is a Socket LAM configured for the same source in two separate clusters. Other groups include the following;

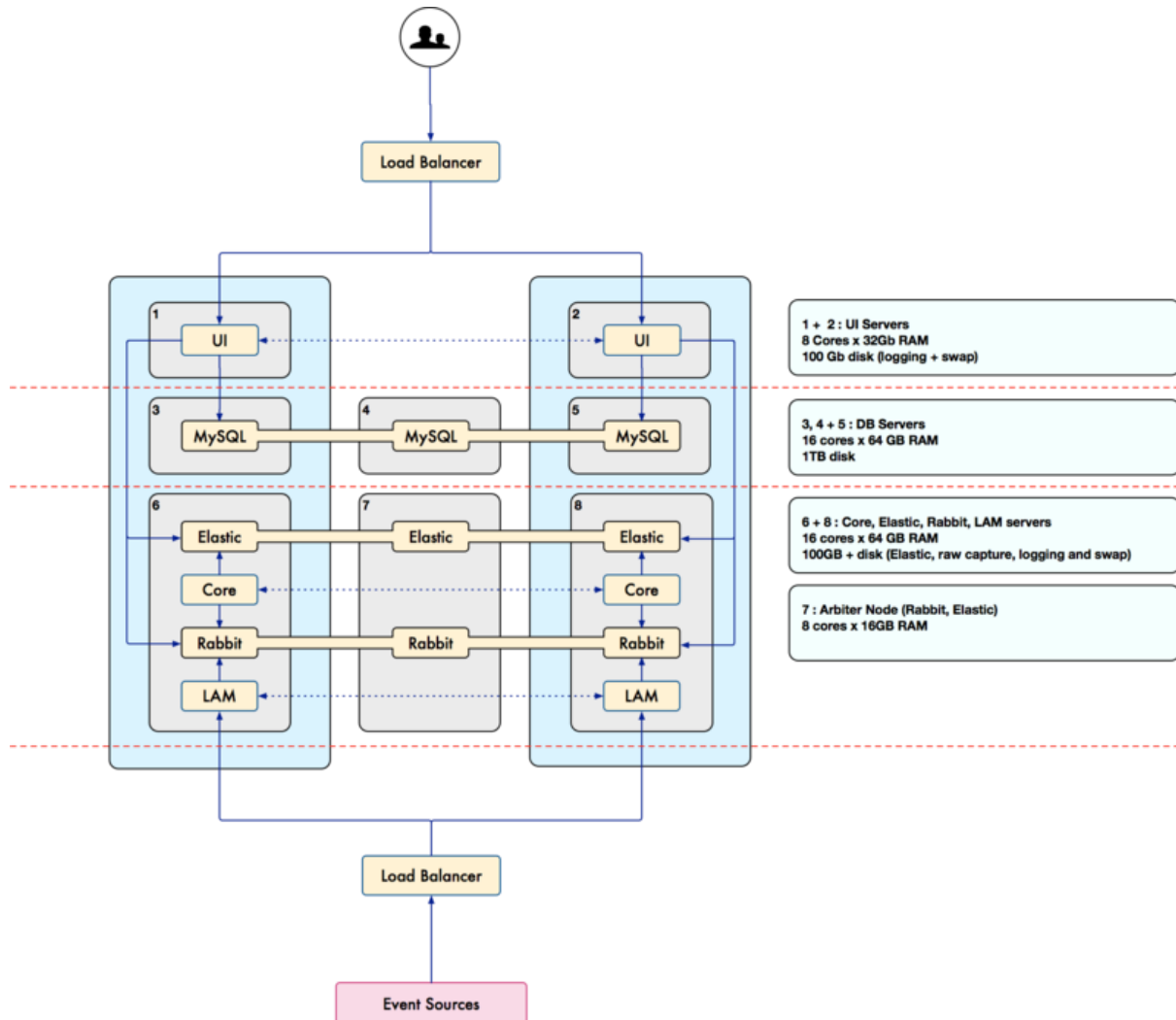
- Servlets for the UI.
- Moogfarmd for data processing.
- Individual LAMs for data ingestion. For example the REST LAM.

An **instance** is an individual component running within a group. Each instance in a group provides resilience for the other instance. For example the primary instance of a Socket LAM pairs with a secondary instance in the second cluster to make a group.



High Availability Reference Architecture

This diagram in this topic represents a Cisco Crosswork Situation Manager High Availability deployment to a single site: one datacenter, LAN, or availability zone. To support this architecture, all servers must have sufficient connection speed amongst themselves so that latency between hosts does not exceed 5 ms.



A) Load Balancers / VIPs

All of components Cisco Crosswork Situation Manager have their own HA mechanism that provides failover capabilities , but it is also a best practice to use a load balancer or load balancers. Load balancers can be hardware or software based with the following requirements and recommendations:

- Load balancers must use TCP.
- You must implement health checks using your preferred approach to remove unhealthy servers from the cluster.



- The load balancer should provide load balancing capabilities and a VIP for each server role. For example: one UI VIP per site, one LAM vip per site.
- Sticky sessions are recommended.
- You can choose your preferred load balancing approach. For example, round robin or least-connection.

B) User Interface

The Cisco Crosswork Situation Manager UI comprises the following components:

- **Nginx**: the web server that provides static UI content and acts as a proxy for the application server. For HA deployments, install a minimum of two Nginx instances on separate servers and optionally cluster the Nginx instances.
- **Tomcat** is the web server that provides servlet and API support. For HA deployments, install a minimum of two Tomcat instances on separate servers and optionally cluster the instances.

Required Ports: 80, 443

C) Data Ingestion

Cisco Crosswork Situation Manager uses the following types of Link Access Modules (LAMs) to ingest data:

- Polling LAMs that periodically connect to a data source using an integration API to collect event data.
- Receiving LAMs that provide an endpoint for data sources to post event data.

For HA deployments:

- Install two instances of each LAM. By default LAMs run in active/passive mode. See </document/preview/77311#UIDaea0bdc20a2d5c12ffc6fce1c1d3b562>.Modes of Operation in High Availability
- For LAMs deployed over an unreliable link such as a WAN, or across data centers, you should deploy a caching LAM strategy that includes a database and message queue on the LAM Servers.
- You can load balance receiving LAMs and configure them as active/active to increase capacity. See </document/preview/77311#UIDaea0bdc20a2d5c12ffc6fce1c1d3b562>.Modes of Operation in High Availability

Required Ports: See

</document/preview/24298#UID3447fd115bb74d43a3b3f8e18db421fe>Integrations Default Ports

D) Core Data Processing



Moogfarmd is the core data processing application that controls all other services in Cisco Crosswork Situation Manager. It manages the clustering algorithms and other applets (Moolets) that run as part of the system. For HA deployments, install a minimum of 2 Moogfarmd services separate servers. Moogfarmd can only run as a two-instance group in an active/passive mode.

Required Ports: 5701, 8901 for Hazelcast: the in-memory data grid that provides fault tolerance.

E) Message Bus

Cisco Crosswork Situation Manager uses RabbitMQ as the system message bus. It requires a minimum of three servers for HA. RabbitMQ relies on its native clustering functionality and mirrored queues to handle failover; it does not use the Cisco Crosswork Situation Manager load balancing feature.

Required Ports: 5672, 4369, 15672

F) Database

Cisco Crosswork Situation Manager uses MySQL as the system database. It requires a minimum of three servers configured in master/standby/standby for HA.

Required Ports: 3306

G) Search and Indexing

Cisco Crosswork Situation Manager uses Elasticsearch to store active alert and Situation data to provide search functionality within the product. For HA deployments install a cluster of a minimum of three data servers with one active master server.

Required Ports: 9200, 9300

[High Availability for Third Party Component Dependencies](#)

You can configure Cisco Crosswork Situation Manager dependencies such as Apache Tomcat, Nginx, MySQL, Grafana, RabbitMQ and Elasticsearch to work effectively in highly available deployments.

See [High Availability](#) (HA) for details on high availability deployments of Cisco Crosswork Situation Manager and deployment scenarios.

Configure Apache Tomcat for HA

You can set up Apache Tomcat for high availability using multiple Tomcat servers that you can cluster.

See [High Availability Reference Architecture](#) for an example. Refer to Apache's documentation about [Tomcat Clustering](#).

Configure Nginx for HA



You can configure Nginx to run on multiple servers in a high-availability cluster to remove any single points of failure in your distributed deployment.

See [High Availability Reference Architecture](#) for an example. Refer to Nginx's documentation about [High Availability](#).

Configure MySQL for HA

You can configure MySQL servers in a master-slave setup to ensure your distributed deployment of Cisco Crosswork Situation Manager is highly available.

See [High Availability Reference Architecture](#) for an example. Refer to MySQL's documentation about [High Availability](#).

Configure RabbitMQ for HA

You can improve the performance and reliability of your Cisco Crosswork Situation Manager deployment by:

- Distributing your RabbitMQ brokers on different hosts.
- Clustering your multiple RabbitMQ brokers.
- Mirror your message queues across multiple nodes.

See [Message System Deployment](#) for setup steps. Refer to the RabbitMQ documentation on [Clustering](#) and [Mirrored Queues](#).

Configure Elasticsearch for HA

There are different ways to configure Elasticsearch for distributed installations. See [High Availability Reference Architecture](#) for more information.

Refer to the Elasticsearch documentation about [Clustering](#) for more details.

Configure Grafana for HA

To configure Grafana for distributed installations, you should configure each Grafana instance to connect to a Cisco Crosswork Situation Manager UI load balancer such as HAProxy rather than the Cisco Crosswork Situation Manager UI stack.

Alternatively you can point it at the Apache Tomcat server or Nginx server. Refer to the Grafana documentation on [Setting Up Grafana for High Availability](#).

[Single Site High Availability](#)

This topic guides you through the most basic high availability setup for Cisco Crosswork Situation Manager where you install two clusters, one primary and one secondary. It covers the configuration for the core Cisco Crosswork Situation Manager components as follows:

- How to designate multiple hosts for the message bus and search components.
- How to configure system failover options.



- How to configure the cluster for the overall system, Moogfarmd, and LAMs.

These instructions don't include instructions for setting up load balancing or other infrastructure topics. For HA for third-party components: RabbitMQ, Elasticsearch, and MySQL, see [High Availability for Third Party Component Dependencies](#).

Note

For the sake of brevity and clarity the sample configurations in this document don't include the default comments delivered in the file.

Before You Begin

Before you start your highly available deployment of Moogsoft AIOps:

- Familiarize yourself with the single-server deployment process: [Install and Upgrade Cisco Crosswork Situation Manager](#).
- Read the [High Availability Overview](#) and review the [High Availability Reference Architecture](#).
- Set up Cisco Crosswork Situation Manager on three hosts following the standard installation instructions. The example here uses the host1 and host2 and host3 as example machine names. On the third host, only initialize the message bus, search, and the database.

Tip

If you are using virtual hosts to perform your HA setup, you can setup Moogsoft AIOps and perform the HA configurations on the first host. Then clone your first host to create the second host and third host. Then you only need to change the configurations that differ on the other hosts.

- Verify that both hosts can access the required ports on the other host in the pair. See [High Availability Reference Architecture](#).
- Set up HA for RabbitMQ, Elasticsearch, and MySQL. See [Cisco Crosswork Situation Manager](#).

Configure System Level HA Settings

Edit the system-wide settings to configure HA in the following file: `MOOGSOFT_HOME/config/system.conf`. For details on `system.conf`, see [System Configuration Reference](#).

1. Edit the message bus (`mooms`) brokers object to include all three hosts. The setting is the same on both `host1` and `host2`. For example:

```
"mooms" :
{
  "zone" : "MY_ZONE",
  "brokers" : [
```



```
{
  "host"      : "host1",
  "port"     : 5672},
{
  "host"      : "host2",
  "port"     : 5672 },
{
  "host"      : "host3",
  "port"     : 5672 },
],
```

2. Edit the database (mysql) object to target one of the following:

- the master database. This is OK for testing HA, but requires a manual failover to the standby database server. a vip for the MySQL database.
- a vip or a load balancer for the MySQL database.

The setting is the same on both host1 and host2. For example, if you target the master database on host1:

```
"mysql" :
{
  "host"          : "host1",
  "moogdb_database_name"      : "moogdb",
  "referencedb_database_name" : "moog_reference",
  "username"       : "ermintrude",
  # "encrypted_password": "vQj7/yom7e5ensSEb10v2Rb/pgkaPK
/40cU1EjYNtQU=
",
  "password"      : "m00",
  "port"          : 3306
```

3. Edit the search object to include the host and port configuration for all Elasticsearch hosts. The setting is the same for both host1 and host2. For example:

```
"search" :
{
  "connection_timeout" : 1000,
  "request_timeout"    : 10000,
  "limit": 1000,
  "nodes" : [
    {
      "host"      : "host1",
      "port"      : 9200
    },
    {
      "host"      : "host2",
      "port"      : 9200
    },
    {
      "host"      : "host3",
```



```
        "port"      : 9200
      }
    ]
  },
```

4. The failover settings are the same for both host1 and host2. Configure the failover object as follows:
 - Set `persist_state` to true.
 - Optionally set `auto_increment` to true if you want Hazelcast to attempt to automatically increment the port number to find an available port in the case that 5701 is not available.
 - Add both hosts where you are running Moogfarmd and LAMs to the hosts lists. For example host1 and host2.
 - Disable the `man_center` UI for Hazelcast.
 - Set `cluster_per_group` to true.
 - Set `automatic_failover` to true. Otherwise, you must use the HA Control utility, `ha_cntl`, to manually trigger failover in the event of a component failure.
 - In most cases, the default settings for Moogfarmd failover work fine. For noisy environments, you can increase the margin value to 30. For more information on individual settings, see [In most cases, the default settings for Moogfarmd failover work fine. For noisy environments, you can increase the margin value to 30. For more information on individual settings, see \[System Configuration Reference\]\(#\).](#)

For example:

```
"failover" :
{
  "persist_state" : true,
  "hazelcast" :
  {
    "network_port"
    "auto_increment"
    "hosts" : ["host1, host2"],
    "man_center" :
      {
        "enabled" : false,
        "host" : "localhost",
        "port" : 8091
      },
    "cluster_per_group" : true
  },
  "keepalive_interval" : 5,
  "margin" : 10,
```



```
    "failover_timeout" : 10,  
    "automatic_failover" : true,  
    "heartbeat_failover_after": 2  
  },
```

5. Configure the cluster name. This setting differs from host to host. For example on host1:

```
  "ha": {  
    "cluster": "host1"  
  },
```

6. Edit the web host name. Enter the domain name or vip of the web server load balancer. For example, if your domain for Cisco Crosswork Situation Manager is myaiops.example.com:

```
  "webhost" : myaiops.example.com
```

Enable HA for Servlets

Edit the servlets settings to configure HA in the following file:
MOOGSOFT_HOME/config/system.conf.

Uncomment the HA object and edit the HA properties. This vary from cluster to cluster based upon your instance names. Both instances are active, so set start_as_passive to false. For example, on host1:

```
,  
  ha :  
  {  
    #cluster: "MOO",  
    instance: "servlets-host_",  
    group: "UI",  
    start_as_passive: false  
  }
```

You don't need to edit the cluster name, the servlets inherit it from system.conf.

Enable HA for Moogfarmd

Edit the Moogfarmd settings in the following file:
\$MOOGSOFT_HOME/config/moogfarmd.conf. The configurations vary from cluster to cluster based upon your instance names.

Uncomment the HA object and edit the HA properties. For example on host1:

```
,  
  ha:  
  {  
    cluster: "MOO",  
    group: "moog_farmd",  
    instance: "moogfarmd_host1",  
    default_leader: true,
```



```
        start_as_passive: false
    ]
```

Enable HA for LAMs

Each LAM has its own configuration file under `MOOGSOFT_HOME/config/`. This example references `rest_lam.conf`.

These instructions apply to the the default active / passive mode. Uncomment the HA object and edit the HA properties. This will vary from cluster to cluster based upon your instance names. Cisco Crosswork Situation Manager automatically manages the active and passive role for the LAMs in a single process group. For example on host1:

```
,
  ha:
    {
      group: "rest_lam",
      instance: "rest_lam_host1",
      duplicate_source: false
    },
```

Restart Components

After you have finished your configurations, restart the Cisco Crosswork Situation Manager on

[HA Control Command Reference](#)

TBD : written just need migration

Tarball Installation and Upgrade

You can install and upgrade Cisco Crosswork Situation Manager using a tarball. Benefits of performing a tarball installation include:

- You can perform the installation as a user other than root. The installation steps also work for the root user. Be sure to perform all steps as the same user.
- You can install Cisco Crosswork Situation Manager to the directory of your choice.
- You do not need to use a package manager (RPM) to install Cisco Crosswork Situation Manager.

The available tarball installation and upgrade options are as follows:

If you want to install Cisco Crosswork Situation Manager as the root user, see [RPM Installation and Upgrade](#). For other installation and upgrade options see [Install and Upgrade Moogsoft AIOps](#).

Distributed Tarball Installation

You can install the different of Cisco Crosswork Situation Manager packages on multiple machines in order to distribute the workload. Dividing the workload for Cisco



Crosswork Situation Manager components between multiple machines can provide performance improvement, but it does not provide redundancy or failover. For those features, see [High Availability Overview](#).

This topic guides you through the most common distributed installation scenario, which is to run the MySQL database on one host and all other processes on a separate host.

Before You Begin

Before you run the Cisco Crosswork Situation Manager installation, perform the following verification and preparation steps:

- Identify the CentOS 7 hosts where you plan run your distributed installation. The example uses server1 for the database host and server2 for the host for all other processes.
- Identify the Linux user you want to use for installation. Use the same user credentials later if for any reason you need to start or stop Cisco Crosswork Situation Manager processes.
- Optionally set the ulimit maximum for open files and max user processes for the installation user. For example, on a busy system you could increase both to 65535. Setting ulimit values requires root permissions.
- Choose a working directory to run your installation. The examples in this document use the current user's home folder, ~, as the working directory. The installation directory requires a minimum of 7GB. If you are also storing the MySQL database in the installation directory, you'll require more space to allow the growth of log files and storage of artefacts in the database and Elasticsearch.
- Verify ports 8443 and 8080 are open. If you are installing as root, Verify ports 443 and 80 are open.
- Ensure both servers can communicate using the MySQL port: 3306 by default.
- Verify your system is running OpenSSL v1.02 or later.

Install Cisco Crosswork Situation Manager and MySQL

Install Cisco Crosswork Situation Manager and MySQL on both hosts, for example server1 and server2. The database server uses some components from Cisco Crosswork Situation Manager and Cisco Crosswork Situation Manager uses the database client.

Perform the following installation steps on both hosts unless otherwise noted:

1. Download the Cisco Crosswork Situation Manager non-root installer. You can download this via a web browser from <https://speedy.moogsoft.com/installer/> and use the yum user credentials provided by Cisco support. Alternatively you can update the following command with your user credentials to download the installer:



```
curl -L -O
"https://<username>:<password>@speedy.moogsoft.com/installer/moogsoft-
aiops-7.2.0.tgz"
```

2. Unzip and untar the distribution archive in your working directory:

```
tar -xf moogsoft-aiops-7.2.0.tgz
```

3. Install MySQL mysql-community-server-5.7.22. For example to install to your working directory:

- Download the mysql-community-server-5.7.22 distribution package to your working directory:

```
curl -L -O https://dev.mysql.com/get/Downloads/MySQL-5.7/mysql-
5.7.22-el7-x86_64.tar.gz
```

- Unzip and untar the MySQL distribution package:

```
tar -xf mysql-5.7.22-el7-x86_64.tar.gz
```

- Add the MySQL binary path to your \$PATH environment variable. For instance, using the admin user's home folder:

```
echo "export PATH=$PATH:~/mysql-5.7.22-el7-x86_64/bin" >>
~/.bashrc && \
source ~/.bashrc
```

4. Install Kernel Asynchronous I/O (AIO) Support for Linux. For example:

```
mkdir -p ~/install/libraries/ && cd ~/install/libraries/ && \
curl -L -O http://mirror.centos.org/centos/7/os/x86_64/Packages/libaio-
0.3.109-13.el7.x86_64.rpm && \
rpm2cpio ./libaio-0.3.109-13.el7.x86_64.rpm | cpio -idmv && \
rm -f ./libaio-0.3.109-13.el7.x86_64.rpm && \
rm -f ~/install/libraries/lib64/libaio.so.1 && \
ln -s ~/install/libraries/lib64/libaio.so.1.0.1
~/install/libraries/lib64/libaio.so.1 && \
echo "export LD_LIBRARY_PATH=`pwd`/lib64:\$LD_LIBRARY_PATH" >>
~/.bashrc && \
source ~/.bashrc && \
cd -
```

5. Install libgfortran v4.4.6 or later on the host that runs the Cisco Crosswork Situation Manager components; not the database host. For example on server2:

```
mkdir -p ~/install/libraries/ && cd ~/install/libraries/ && \
curl -L -O
http://mirror.centos.org/centos/7/os/x86_64/Packages/libquadmath-4.8.5-
36.el7.x86_64.rpm && \
rpm2cpio ./libquadmath-4.8.5-36.el7.x86_64.rpm | cpio -idmv && \
rm ./libquadmath-4.8.5-36.el7.x86_64.rpm && \
```



```
curl -L -O
http://mirror.centos.org/centos/7/os/x86_64/Packages/libgfortran-4.8.5-
36.el7.x86_64.rpm && \
rpm2cpio ./libgfortran-4.8.5-36.el7.x86_64.rpm | cpio -idmv && \
rm ./libgfortran-4.8.5-36.el7.x86_64.rpm && \
echo "export LD_LIBRARY_PATH=`pwd`/usr/lib64:\$LD_LIBRARY_PATH" >>
~/.bashrc && \
source ~/.bashrc && \
cd -
```

6. Execute the installation script in your working directory to install Cisco Crosswork Situation Manager.

```
bash moogsoft-aiops-install-7.2.0.sh
```

The script guides you through the installation process and lets you choose the installation directory, by default <working-directory>/moogsoft.

7. Set the \$MOOGSOFT_HOME environment variable to point to your installation directory. For example:

```
echo "export MOOGSOFT_HOME=/home/admin/moogsoft" >> ~/.bashrc
echo "export PATH=$PATH:$MOOGSOFT_HOME/bin/utils" >> ~/.bashrc && \
source ~/.bashrc
```

Initialize the Database Server

Initialize the database server first so it will be available for dependent processes when they start up.

To set up the database, execute the database initialization script `moog_init_db.sh` on the database host. For example on `server1`:

```
$MOOGSOFT_HOME/bin/utils/moog_init_db.sh -Iu root
```

Add MySQL 'grants' to allow **SERVER2** to communicate with the **SERVER1** MySQL instance (please substitute <**SERVER2_IP_ADDRESS**> with the IP address of **SERVER2**):

```
mysql -u root -S $MOOGSOFT_HOME/cots/mysql/var/lib/mysql/mysql.sock <<<
"GRANT ALL PRIVILEGES ON *.* TO 'root'@'<SERVER2_IP_ADDRESS>' IDENTIFIED BY
'' with grant option;"
mysql -u root -S $MOOGSOFT_HOME/cots/mysql/var/lib/mysql/mysql.sock <<<
"GRANT ALL ON moogdb.* TO 'ermintrude'@'<SERVER2_IP_ADDRESS>' IDENTIFIED BY
'm00'; \
```

```
GRANT ALL ON moog_reference.* TO ermintrude@'<SERVER2_IP_ADDRESS>'
IDENTIFIED BY 'm00'; \
```

```
GRANT ALL ON historic_moogdb.* TO 'ermintrude'@'<SERVER2_IP_ADDRESS>'
IDENTIFIED BY 'm00';"
mysql -u root -S $MOOGSOFT_HOME/cots/mysql/var/lib/mysql/mysql.sock <<<
"GRANT ALL ON moogdb.* TO 'root'@'<SERVER2_IP_ADDRESS>' IDENTIFIED BY '';
```



```
GRANT ALL ON moog_reference.* TO root@'<SERVER2_IP_ADDRESS>' IDENTIFIED BY  
'';\
```

```
GRANT ALL ON historic_moogdb.* TO root@'<SERVER2_IP_ADDRESS>' IDENTIFIED BY  
'';"
```

Initialize Cisco Crosswork Situation Manager

After the database is available, you can initialize the remaining Cisco Crosswork Situation Manager processes. For example on host2:

1. Configure the toolrunner to execute locally. Edit `$MOOGSOFT_HOME/config/servlets.conf`.
2. Uncomment the `execute_locally` property, and value set the value to 'true' as follows. Take special note of the initial comma and the absence of the trailing comma.

```
, execute_locally: true
```

3. Execute the Cisco Crosswork Situation Manager initialization script, `moog_init.sh`, to set up the remaining Cisco Crosswork Situation Manager processes.

```
$MOOGSOFT_HOME/bin/utils/moog_init.sh -qI MoogsoftAIOps -u root --  
accept-eula -d <server1_ip>:3306
```

Where `<server1_ip>` is the IP address of the database server.

4. Run the following command to start Moogfarmd:

```
$MOOGSOFT_HOME/bin/utils/process_cntl moog_farmd start
```

After Moogfarmd starts, log in to Cisco Crosswork Situation Manager at the following URL: `https://<server1_ip>:8443`

Single Host Tarball Installation

This topic covers the procedure to install Cisco Crosswork Situation Manager for the following scenarios:

- You can perform the installation as the root user or as a user other than root. Be sure to perform all steps as the same user.
- You can install Cisco Crosswork Situation Manager to the directory of your choice.
- You do not need to use a package manager (RPM) to install Cisco Crosswork Situation Manager .

Before You Begin

Before you run the Cisco Crosswork Situation Manager installation, perform the following verification and preparation steps:



1. Choose a CentOS 7 / RHEL 7 system.

Note

These instructions are for a new installation and assume you have not previously installed Cisco Crosswork Situation Manager. If you have defined Moogsoft environment variables such as `$MOOGSOFT_HOME` for the installation user, remove the environment variables before continuing the installation.

2. Identify the Linux user you want to use for installation. Use the same user credentials later if for any reason you need to start or stop Cisco Crosswork Situation Manager processes.
3. Optionally set the ulimit maximum for open files and max user processes for the installation user. For example, on a busy system you could increase both to 65535. Setting ulimit values requires root permissions.
4. Choose a working directory to run your installation. The examples in this document use `/home/<your_username>` as the working directory. The installation directory requires a minimum of 7GB. If you are also storing the MySQL database in the installation directory, you will require more space to allow the growth of log files and storage of artifacts in the database and Elasticsearch.
5. Download the Cisco Crosswork Situation Manager non-root installer. You can download this via a web browser from <https://speedy.moogsoft.com/installer/> and use the yum user credentials provided by Cisco support. Alternatively you can update the following command with your user credentials to download the installer:

```
curl -L -O
"https://<username>:<password>@speedy.moogsoft.com/installer/moogsoft-
aiops-7.2.0.tgz"
```

6. Unzip and untar the distribution archive in your working directory:

```
tar -xf moogsoft-aiops-7.2.0.tgz
```

The distribution archive `moogsoft-aiops-<version>.tgz` contains:

- A README.txt file
 - The installation script: `moogsoft-aiops-install-7.2.0.sh`
 - A Cisco Crosswork Situation Manager archive: `moogsoft-aiops-dist-7.2.0.tgz`
 - An integrations archive: `moogsoft-aiops-integrations-dist-7.2.0.tgz`
7. Install MySQL `mysql-community-server-5.7.22`. For example to install to your working directory:
 - Download the `mysql-community-server-5.7.22` distribution package to your working directory:



```
curl -L -O https://dev.mysql.com/get/Downloads/MySQL-5.7/mysql-5.7.22-el7-x86_64.tar.gz
```

- Unzip and untar the MySQL distribution package:

```
tar -xf mysql-5.7.22-el7-x86_64.tar.gz
```

- Add the MySQL binary path to your \$PATH environment variable.

```
echo "export PATH=$PATH:~/mysql-5.7.22-el7-x86_64/bin" >> \
~/.bashrc && \
source ~/.bashrc
```

For instance, using the /home/admin working directory:

```
echo "export PATH=$PATH:/home/admin/mysql-5.7.22-el7-x86_64/bin" \
>> ~/.bashrc && \
source ~/.bashrc
```

8. Install Kernel Asynchronous I/O (AIO) Support for Linux. For example:

```
mkdir -p ~/install/libraries/ && cd ~/install/libraries/ && \
curl -L -O http://mirror.centos.org/centos/7/os/x86_64/Packages/libaio-0.3.109-13.el7.x86_64.rpm && \
rpm2cpio ./libaio-0.3.109-13.el7.x86_64.rpm | cpio -idmv && \
rm -f ./libaio-0.3.109-13.el7.x86_64.rpm && \
rm -f ~/install/libraries/lib64/libaio.so.1 && \
ln -s ~/install/libraries/lib64/libaio.so.1.0.1 \
~/install/libraries/lib64/libaio.so.1 && \
echo "export LD_LIBRARY_PATH=`pwd`/lib64:~$LD_LIBRARY_PATH" >> \
~/.bashrc && \
source ~/.bashrc && \
cd -
```

9. Navigate to your working directory and install libgfortran v4.4.6 or later. For example:

```
cd ~/install/libraries/ && \
curl -L -O \
http://mirror.centos.org/centos/7/os/x86_64/Packages/libquadmath-4.8.5-36.el7.x86_64.rpm && \
rpm2cpio ./libquadmath-4.8.5-36.el7.x86_64.rpm | cpio -idmv && \
rm ./libquadmath-4.8.5-36.el7.x86_64.rpm && \
curl -L -O \
http://mirror.centos.org/centos/7/os/x86_64/Packages/libgfortran-4.8.5-36.el7.x86_64.rpm && \
rpm2cpio ./libgfortran-4.8.5-36.el7.x86_64.rpm | cpio -idmv && \
rm ./libgfortran-4.8.5-36.el7.x86_64.rpm && \
echo "export LD_LIBRARY_PATH=`pwd`/usr/lib64:~$LD_LIBRARY_PATH" >> \
~/.bashrc && \
source ~/.bashrc && \
cd -
```



10. Verify ports 8443 and 8080 are open. If you are installing as root, Verify ports 443 and 80 are open.
11. Verify your system is running OpenSSL v1.0.2 or later.

Install Cisco Crosswork Situation Manager

1. Execute the installation script `moogsoft-aiops-install-7.2.0.sh` in your working directory to install Cisco Crosswork Situation Manager.

The script guides you through the installation process and lets you choose the installation directory, by default `<working-directory>/Cisco`.

```
bash moogsoft-aiops-install-7.2.0.sh
```

2. As a convenience, set the `$MOOGSOFT_HOME` environment variable to point to your installation directory. Additionally, add `$MOOGSOFT_HOME/bin/utils` to the path. For example:

```
echo "export MOOGSOFT_HOME=~/.moogsoft" >> ~/.bashrc
echo "export PATH=$PATH:~/.moogsoft/bin/utils" >> ~/.bashrc && \
source ~/.bashrc
```

Initialize Cisco Crosswork Situation Manager

After you finish the installation process, initialize Cisco Crosswork Situation Manager as follows:

1. Configure the toolrunner to execute locally by setting "execute_locally: true" in `$MOOGSOFT_HOME/config/servlets.conf`:

```
sed -i 's/# execute_locally: false,/,execute_locally: true/1'
$MOOGSOFT_HOME/config/servlets.conf
```

2. Execute the initialization script `startup` and bootstrap your system.

```
$MOOGSOFT_HOME/bin/utils/moog_init.sh -I <zone_name> -u root
```

- The script prompts you to accept the End User License Agreement (EULA) and guides you through the initialization process.
- The script initializes all components on the current host, loads the MySQL schemas using the MySQL user.
- The 'zone name' sets up a virtual host for the message bus. If you have multiple systems sharing the same bus, use a different zone name for each. For example:

```
moog_init.sh -I MoogsoftAIOps -u root
```

- If you would like the processes used by AIOps (MySQL, RabbitMQ, Moogfarmd etc) to be restarted when the system is rebooted, the `moog_init.sh` command above should include a '-k' flag. For example:



```
$MOOGSOFT_HOME/bin/utils/moog_init.sh -kI <zone_name> -u root
```

For more information on this feature please see: [Configure Services to Restart](#)

Unattended Installation Example

The `moog_init` script provides the capability to run a quiet install and to automatically accept the EULA. This means you can write a bash script to automatically execute both the installation script and the initialization script on the same host. For example:

```
~/moogsoft-aiops-install-7.2.0.sh \  
-d ~/moogsoft &&  
  
~/moogsoft/bin/utils/moog_init.sh \  
-qI MoogsoftAIOps -p MySQLpasswd -u root --accept-eula
```

Run `moog_init.sh -h` for a description of all the flags.

Post-Initialization

Verify your installation by following the steps in [Post Install Validation](#).

You can login to Cisco Crosswork Situation Manager at the following URL:

https://<server_ip/hostname>:8443

You can use the Process Control Utility `process_cntl` to start, stop, and check the status of the processes. See [Control Moogsoft AIOps Processes](#).

Follow the [Troubleshooting](#) guide to resolve any issues.

Upgrade - Tarball Deployments

This guide provides instruction on how to upgrade to Cisco Crosswork Situation Manager ESR v7.2.0. You must upgrade to v7.1.0 before you upgrade to v7.2.0.

For instructions on how to upgrade from a supported version, refer to the following topics:

- [Upgrade Tarball v6.5.x to Tarball v7.2.x](#)
- [Upgrade Tarball v7.0.x to Tarball v7.2.x](#)
- [Upgrade Tarball v7.1.x to Tarball v7.2.x](#)

For instructions on how to upgrade other releases, see [Moogsoft Releases](#).

Upgrade Tarball v6.5.x to Tarball v7.2.x

This topic describes the upgrade procedure of Tarball deployments from Cisco Crosswork Situation Manager v6.5.x Tarball to Cisco Crosswork Situation Manager v7.2.x Tarball.



Before You Begin

To perform the upgrade you must be logged in as the same non-root user that installed and runs the existing Cisco Crosswork Situation Manager v6.5.x software.

Download the Installer

Run the following command to download the installer:

```
curl -L -O
"https://<username>:<password>@speedy.moogsoft.com/installer/moogsoft-
aiops-7.2.0.tgz"
```

Note UI Integrations Configuration

If you are using any UI integrations, make a note of their connection and configuration details before you begin the upgrade. You will need to delete and reconfigure them in the post-upgrade steps, due to UI changes in the new version.

Stop Services and Processes

1. Stop Moogfarmd:

```
$MOOGSOFT_HOME/bin/utils/process_cntl moog_farmd stop
```

2. Stop Apache Tomcat:

```
$MOOGSOFT_HOME/bin/utils/process_cntl apache-tomcat stop
```

3. Stop the integrations and LAMs, using either the Process Control utility or the kill command:

```
$MOOGSOFT_HOME/bin/utils/process_cntl <lam_name> stop
```

or:

```
kill -9 $(ps -ef | grep java | grep _lam | awk '{print $2}')
2>/dev/null
```

4. Disable the Events Analyser from running during the upgrade process:

- a. Run the following command to comment out the relevant lines in crontab:

```
(crontab -l | sed -e 's/^(.*events_analyser.*)$/#\1/' ) | crontab -
```

- b. Run the following command to stop any active Events Analyser processes:

```
ps -ef | grep java | egrep 'events_analyser|moog_indexer' | awk '{print
$2}' | xargs kill 2>/dev/null
```

5. Stop MySQL, RabbitMQ, Elasticsearch, Nginx and Apache Tomcat:

```
$MOOGSOFT_HOME/bin/utils/process_cntl mysqld stop
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch stop
```



```
$MOOGSOFT_HOME/bin/utils/process_cntl rabbitmq stop
$MOOGSOFT_HOME/bin/utils/process_cntl nginx stop
```

6. Obtain the path variables before re-linking the directories. From this step onwards, use the same terminal session to keep the variables:

```
CERT_REAL_PATH_PEM=$(readlink -f $(grep -h 'ssl_certificate '
$MOOGSOFT_HOME/dist/6.5.*/cots/nginx/config/conf.d/moog-ssl.conf|head -
1|awk '{print $2}'|tr -d ';'))
CERT_REAL_PATH_KEY=$(readlink -f $(grep -h 'ssl_certificate_key'
$MOOGSOFT_HOME/dist/6.5.*/cots/nginx/config/conf.d/moog-ssl.conf|head -
1|awk '{print $2}'|tr -d ';'))
CERT_PATH_PEM=$(grep -h 'ssl_certificate '
$MOOGSOFT_HOME/dist/6.5.*/cots/nginx/config/conf.d/moog-ssl.conf|head -
1)
CERT_PATH_KEY=$(grep -h 'ssl_certificate_key'
$MOOGSOFT_HOME/dist/6.5.*/cots/nginx/config/conf.d/moog-ssl.conf|head -
1)
```

Back Up the Existing System

To back up the existing system:

1. Back up \$MOOGSOFT_HOME.
2. Take a snapshot (for VMs).
3. Back up MySQL.

Upgrade Cisco Crosswork Situation Manager

Run the following commands to perform the upgrade:

```
tar -xf moogsoft-aiops-7.2.0.tgz
bash moogsoft-aiops-install-7.2.0.sh
```

Follow the instructions that appear. The upgrade process detects the v7.1.0 installation and performs an upgrade (a side-by-side deployment).

Migrate Data Directories and Restart MySQL and Elasticsearch

Run the following commands to migrate the data directories for MySQL, RabbitMQ, and Elasticsearch into the new locations:

```
mkdir -p $MOOGSOFT_HOME/var/lib/{elasticsearch,mysql,rabbitmq}
cp -frp $MOOGSOFT_HOME/dist/6.5.*/cots/elasticsearch/data
$MOOGSOFT_HOME/var/lib/elasticsearch/
sed -i "s;#path.data: /path/to/data;path.data:
$MOOGSOFT_HOME/var/lib/elasticsearch/data;g"
$MOOGSOFT_HOME/cots/elasticsearch/config/elasticsearch.yml
cp -frp $MOOGSOFT_HOME/dist/6.5.*/cots/mysql/*
$MOOGSOFT_HOME/var/lib/mysql/
sed -i "s;$MOOGSOFT_HOME/cots/mysql/;$MOOGSOFT_HOME/var/lib/mysql/;"
```



```
~/my.cnf
cp -frp $MOOGSOFT_HOME/dist/6.5.*/cots/rabbitmq-
server/var/lib/rabbitmq/mnesia $MOOGSOFT_HOME/var/lib/rabbitmq/
```

Run the following command to change the default log level for Elasticsearch. By default it is set to INFO but you can change it to WARN to reduce the size of the Elasticsearch logs.

```
sed -i -e "s;rootLogger.level = info;rootLogger.level = warn;g"
$MOOGSOFT_HOME/cots/elasticsearch/config/log4j2.properties
```

Run the following commands to restart the MySQL, RabbitMQ and Elasticsearch processes:

```
$MOOGSOFT_HOME/bin/utils/process_cntl mysqld start
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch start
$MOOGSOFT_HOME/bin/utils/process_cntl rabbitmq start
```

Merge the Latest Configuration File Changes

The top-level \$MOOGSOFT_HOME/config and \$MOOGSOFT_HOME/bots folders are the master folder locations for configuration and bot files. The new 'default' v7.2.0 versions of these files are stored in \$MOOGSOFT_HOME/dist/7.2.0/config/ and \$MOOGSOFT_HOME/dist/7.2.0/bots/ respectively.

1. Identify the config files that have changed between v6.5.0 and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/6.5.*/config
$MOOGSOFT_HOME/dist/7.2.0/config | grep -i 'differ'
```

2. Update the files in \$MOOGSOFT_HOME/config with any changes introduced in the v7.2.0 versions of these files.
3. Identify the contrib files that have changed between v6.5.0 and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/6.5.*/contrib
$MOOGSOFT_HOME/dist/7.2.0/contrib | grep -i 'differ'
```

4. Update the files in \$MOOGSOFT_HOME/contrib with any changes introduced in the v7.2.0 versions of these files.

5. Identify the bot files that have changed between v6.5.0 and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/6.5.*/bots $MOOGSOFT_HOME/dist/7.2.0/bots
| grep -i 'differ'
```

6. Update the files in \$MOOGSOFT_HOME/bots with any changes introduced in the v7.2.0 versions of these files.

Upgrade the Cisco Crosswork Situation Manager Database Schema

To upgrade the Cisco Crosswork Situation Manager database schema, you must provide the Auto Upgrader utility with the credentials of a database user with super privileges.



For single-host installations where MySQL was installed as part of the Cisco Crosswork Situation Manager deployment, you can use the default 'root' user.

1. Run the following command after substituting the <MySQL-SuperUsername> argument:

```
bash $MOOGSOFT_HOME/bin/utils/moog_db_auto_upgrader -t 7.2.0 -u <MySQL-SuperUsername>
```

2. Enter the password for that user. You can provide the password to the utility with the -p flag but this is not recommended in non-test deployments for security reasons.

Drop Deprecated Historic Database Tables

Run the following commands to drop two tables from the historic database that are no longer used. If they are not dropped at this stage, the moog_db_validator utility at the end of this process will report their presence as a delta.

```
bash $MOOGSOFT_HOME/bin/utils/moog_mysql_client -i -e "drop table room_post_sigs";
bash $MOOGSOFT_HOME/bin/utils/moog_mysql_client -i -e "drop table room_posts";
```

Update the RabbitMQ Configuration

Update the RabbitMQ configuration to support autoheal and other processes that improve stability in HA environments. To do this, run the following commands on each server the moogsoft-mooms / RabbitMQ package is installed on. Replace <VHOST> with the name of your RabbitMQ vhost:

```
cp -f $MOOGSOFT_HOME/etc/cots/rabbitmq/rabbitmq.config
$MOOGSOFT_HOME/cots/rabbitmq-server/etc/rabbitmq/
bash $MOOGSOFT_HOME/bin/utils/moog_init_mooms.sh -z <VHOST> -p
```

Upgrade Apache Tomcat and Nginx

Enter the following commands to upgrade Apache Tomcat and Nginx:

```
$MOOGSOFT_HOME/bin/utils/moog_init_ui.sh -tnfwz
$( $MOOGSOFT_HOME/bin/utils/moog_config_reader -k mooms.zone ) --accept-eula
```

Migrate the certificates for the previous deployment of Nginx to the new deployment. Use the following commands as an example of how to do this in the general case where SSL terminates in Nginx (default configuration):

```
cp -f $CERT_REAL_PATH_PEM $MOOGSOFT_HOME/dist/7.2.0/cots/nginx/ssl/
cp -f $CERT_REAL_PATH_KEY $MOOGSOFT_HOME/dist/7.2.0/cots/nginx/ssl/
sed -i "s|.*ssl_certificate .*|${CERT_PATH_PEM}|"
$MOOGSOFT_HOME/dist/7.2.0/cots/nginx/config/conf.d/moog-ssl.conf
sed -i "s|.*ssl_certificate_key .*|${CERT_PATH_KEY}|"
$MOOGSOFT_HOME/dist/7.2.0/cots/nginx/config/conf.d/moog-ssl.conf
```



Restart Nginx:

```
$MOOGSOFT_HOME/bin/utils/process_cntl nginx restart
```

Update and Reconfigure Integrations

This upgrade process replaces the `$MOOGSOFT_HOME/bots/moobots/RemedyIntegration.js` Moobot file. Back up this file if you have customised it, before continuing.

To update the integrations:

1. The LAMs/Integrations no longer use the 'state' files under `$MOOGSOFT_HOME/config`. Remove these files with the following command:

```
rm -f $MOOGSOFT_HOME/config/*_lam.state;
rm -f $MOOGSOFT_HOME/config/{null.state, TestStateManager.state}
rm -f $MOOGSOFT_HOME/config/observe*.conf
```

2. Run the following command to extract the old and new Integrations:

```
bash $MOOGSOFT_HOME/bin/utils/integration_installer -a -l WARN;
for file in $(diff -qr $MOOGSOFT_HOME/dist/7.2.0/ui/integrations/ \
$MOOGSOFT_HOME/dist/6.5.*ui/integrations/ | egrep -vi 'only
in.*7.2.0|differ' \
| awk '{print $NF}'); do cp -rp
$MOOGSOFT_HOME/dist/6.5.*ui/integrations/$file \
$MOOGSOFT_HOME/dist/7.2.0/ui/integrations/; done
```

Configure the ServiceNow MID Server to use Java 8

If you are using a ServiceNow MID server installed on the same host as Cisco Crosswork Situation Manager, you must configure it to point to Java 8. The MID server requires Java 8 (update 152 or later). It will not work with Java 9+. To do this:

1. Install the latest version of Java 8. See the [ServiceNow MID server system requirements](#) for more information.
2. Stop the MID server by running the appropriate command. For example:

```
kill -9 $(ps -ef | grep mid_server | grep -v grep | awk '{print $2}')
```

3. Configure the `wrapper.java.command` property to point to the Java 8 binary in the following file:

```
/usr/local/servicenow/moog_mid_server/agent/conf/wrapper-override.conf.
```

For example:

```
wrapper.java.command=/usr/java/jre1.8.0_171-amd64/bin/java
```

In the next step you will reconfigure the ServiceNow integration which will restart the MID server.



Reconfigure your UI Integrations

Delete and reconfigure all of your UI integrations, using the configuration details you noted prior to the upgrade. This is required due to UI changes in the new version.

Confirm All Processes are Running and Start Moogfarmd

1. Run the following commands to verify that all required processes are running and start Moogfarmd:

```
$MOOGSOFT_HOME/bin/utils/process_cntl apache-tomcat status
$MOOGSOFT_HOME/bin/utils/process_cntl moog_farmd start
$MOOGSOFT_HOME/bin/utils/process_cntl nginx status
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch status
```

2. Run the following command to re-enable the Event Analyser cronjobs:

```
(crontab -l | sed -e 's/^#\+\.(*events_analyser.*\)/\1/' | crontab -
```

- Run the following command to restart any previously running UI-based Integrations:

```
bash $MOOGSOFT HOME/bin/utils/startup cntl;
```

Add the Keepalive Cronjob (Optional)

A new cronjob is installed for clean installations of 7.2.0 which provides the ability for certain services to be restarted automatically should they fail. For RPM deployments this is usually handled by chkconfig or MySQL itself for example, but can be useful in certain cases. See [Configure Services to Restart](#) for more information.

Command to enable if required:

[illegible]

Migrate the Java Keystore

The upgrade includes a new version of the Java Runtime, so you need to migrate any certificates stored in the old Java keystore into the new keystore, which is now the JRE under `$MOOGSOFT_HOME/cots/jre/`.

The previous JRE is located in `$MOOGSOFT_HOME/dist/6.5.0/cots/jre`.

If you did not manually add certificates to the old store, you can skip this step.

Verify the Upgrade

You can verify the upgrade manually or using automatic utilities.

Verify the Upgrade Manually

Perform the following basic steps to ensure the upgrade to Cisco Crosswork Situation Manager v7.2.0 was successful:



1. Verify that the UI login page displays "Version 7.2.0" at the top.
2. Check that the UI "Support Information" window correctly indicates the current version as "7.2.0" and shows the correct schema upgrade history.

Verify the Upgrade Using Automatic Utilities

Run the following automatic utilities to ensure that the upgrade to 7.2.0 was successful:

1. Confirm that all Cisco Crosswork Situation Manager files have been deployed correctly within \$MOOGSOFT_HOME using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_install_validator.sh
```

2. Confirm that all Apache Tomcat files have been deployed correctly within \$MOOGSOFT_HOME using this utility:

```
$MOOGSOFT_HOME/bin/utils/tomcat_install_validator.sh
```

3. Verify that all the steps have completed successfully. If there are some web app differences, run the following command to extract the web app with the correct files:

```
$MOOGSOFT_HOME/bin/utils/moog_init_ui.sh -w
```

4. Confirm that the database schema has upgraded successfully using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_db_validator.sh
```

Confirm that all the steps are successful. Some schema differences might be valid (e.g. custom_info related). If there are more substantial differences, you should investigate further to verify that all the pre-requisite upgrade scripts have been applied in the right order:

```
mysql -u root <moogdb_database_name> -e "select * from  
schema_upgrades;"  
mysql -u root <moog_reference_database_name> -e "select * from  
schema_upgrades;"
```

Remove the MySQL Connector

After upgrading to CCSM 7.2 go to all your machines and remove the file:

```
$MOOGSOFT_HOME/lib/cots/nonDist/mysql-connector-java-  
5.1.45.jar
```

Troubleshooting

If you have any issues, refer to [Troubleshooting](#).

Upgrade Tarball v7.0.x to Tarball v7.2.x

This topic describes the upgrade procedure of Tarball deployments from Cisco Crosswork Situation Manager v7.0.x Tarball to Cisco Crosswork Situation Manager v7.2.x Tarball.



Before You Begin

To perform the upgrade you must be logged in as the same non-root user that installed and runs the existing Cisco Crosswork Situation Manager v7.0.x software.

Download the Installer

Run the following command to download the installer:

```
curl -L -O
"https://<username>:<password>@speedy.moogsoft.com/installer/moogsoft-
aiops-7.2.0.tgz"
```

Note UI Integrations Configuration

If you are using any UI integrations, make a note of their connection and configuration details before you begin the upgrade. You will need to delete and reconfigure them in the post-upgrade steps, due to UI changes in the new version.

Stop Services and Processes

1. Stop Moogfarmd:

```
$MOOGSOFT_HOME/bin/utils/process_cntl moog_farmd stop
```

2. Stop Apache Tomcat:

```
$MOOGSOFT_HOME/bin/utils/process_cntl apache-tomcat stop
```

3. Stop the integrations and LAMs, using either the Process Control utility or the kill command:

```
$MOOGSOFT_HOME/bin/utils/process_cntl <lam_name> stop
```

or:

```
kill -9 $(ps -ef | grep java | grep _lam | awk '{print $2}')
2>/dev/null
```

4. Disable the Events Analyser from running during the upgrade process:

- a. Run the following command to comment out the relevant lines in crontab:

```
(crontab -l | sed -e 's/^(.*events_analyser.*)$/#\1/' ) | crontab -
```

- b. Run the following command to stop any active Events Analyser processes:

```
ps -ef | grep java | egrep 'events_analyser|moog_indexer' | awk '{print
$2}' | xargs kill 2>/dev/null
```

5. Stop MySQL, RabbitMQ, Elasticsearch, Nginx and Apache Tomcat:

```
$MOOGSOFT_HOME/bin/utils/process_cntl mysqld stop
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch stop
```



```
$MOOGSOFT_HOME/bin/utils/process_cntl rabbitmq stop
$MOOGSOFT_HOME/bin/utils/process_cntl nginx stop
```

6. Obtain the path variables before re-linking the directories. From this step onwards, use the same terminal session to keep the variables:

```
CERT_REAL_PATH_PEM=$(readlink -f $(grep -h 'ssl_certificate '
$MOOGSOFT_HOME/dist/7.0.*/cots/nginx/config/conf.d/moog-ssl.conf|head -
1|awk '{print $2}'|tr -d ';'))
CERT_REAL_PATH_KEY=$(readlink -f $(grep -h 'ssl_certificate_key'
$MOOGSOFT_HOME/dist/7.0.*/cots/nginx/config/conf.d/moog-ssl.conf|head -
1|awk '{print $2}'|tr -d ';'))
CERT_PATH_PEM=$(grep -h 'ssl_certificate '
$MOOGSOFT_HOME/dist/7.0.*/cots/nginx/config/conf.d/moog-ssl.conf|head -
1)
CERT_PATH_KEY=$(grep -h 'ssl_certificate_key'
$MOOGSOFT_HOME/dist/7.0.*/cots/nginx/config/conf.d/moog-ssl.conf|head -
1)
```

Back Up the Existing System

To back up the existing system:

1. Back up \$MOOGSOFT_HOME.
2. Take a snapshot (for VMs).
3. Back up MySQL.

Upgrade Cisco Crosswork Situation Manager

Run the following commands to perform the upgrade:

```
tar -xf moogsoft-aiops-7.2.0.tgz
bash moogsoft-aiops-install-7.2.0.sh
```

Follow the instructions that appear. The upgrade process detects the v7.0.x installation and performs an upgrade (a side-by-side deployment).

Run the following command to change the default log level for Elasticsearch. By default it is set to INFO but you can change it to WARN to reduce the size of the Elasticsearch logs.

```
sed -i -e "s;rootLogger.level = info;rootLogger.level = warn;g"
$MOOGSOFT_HOME/cots/elasticsearch/config/log4j2.properties
```

Run the following commands to restart the MySQL, RabbitMQ and Elasticsearch processes:

```
$MOOGSOFT_HOME/bin/utils/process_cntl mysqld start
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch start
$MOOGSOFT_HOME/bin/utils/process_cntl rabbitmq start
```



Merge the Latest Configuration File Changes

The top-level \$MOOGSOFT_HOME/config and \$MOOGSOFT_HOME/bots folders are the master folder locations for configuration and bot files. The new 'default' v7.2.0 versions of these files are stored in \$MOOGSOFT_HOME/dist/7.2.0/config/ and \$MOOGSOFT_HOME/dist/7.2.0/bots/ respectively.

1. Identify the config files that have changed between v7.0.x and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/7.0.*/config  
$MOOGSOFT_HOME/dist/7.2.0/config | grep -i 'differ'
```

2. For the files highlighted by the command above, the differences and new properties in the 7.2.0 versions of these files need to be manually merged into the files in \$MOOGSOFT_HOME/config. Do not overwrite the files in \$MOOGSOFT_HOME/config.

3. Identify the contrib files that have changed between v7.0.x and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/7.0.*/contrib  
$MOOGSOFT_HOME/dist/7.2.0/contrib | grep -i 'differ'
```

4. For the files highlighted by the command above, the differences and new properties in the 7.2.0 versions of these files need to be manually merged into the files in \$MOOGSOFT_HOME/contrib. Do not overwrite the files in \$MOOGSOFT_HOME/contrib.

5. Identify the bot files that have changed between v7.0.x and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/7.0.*/bots $MOOGSOFT_HOME/dist/7.2.0/bots  
| grep -i 'differ'
```

6. For the files highlighted by the command above, the differences and new properties in the 7.2.0 versions of these files need to be manually merged into the files in \$MOOGSOFT_HOME/bots. Do not overwrite the files in \$MOOGSOFT_HOME/bots.

Upgrade the Cisco Crosswork Situation Manager Database Schema

To upgrade the Cisco Crosswork Situation Manager database schema, you must provide the Auto Upgrader utility with the credentials of a database user with super privileges. For single-host installations where MySQL was installed as part of the Cisco Crosswork Situation Manager deployment, you can use the default 'root' user.

1. Run the following command after substituting the <MySQL-SuperUsername> argument:

```
bash $MOOGSOFT_HOME/bin/utils/moog_db_auto_upgrader -t 7.2.0 -u <MySQL-SuperUsername>
```

2. Enter the password for that user. You can provide the password to the utility with the -p flag but this is not recommended in non-test deployments for security reasons.



Drop Deprecated Historic Database Tables

Run the following commands to drop two tables from the historic database that are no longer used. If they are not dropped at this stage, the moog_db_validator utility at the end of this process will report their presence as a delta.

```
bash $MOOGSOFT_HOME/bin/utils/moog_mysql_client -i -e "drop table
room_post_sigs";
bash $MOOGSOFT_HOME/bin/utils/moog_mysql_client -i -e "drop table
room_posts";
```

Update the RabbitMQ Configuration

Update the RabbitMQ configuration to support autoheal and other processes that improve stability in HA environments. To do this, run the following commands on each server the moogsoft-mooms / RabbitMQ package is installed on. Replace <VHOST> with the name of your RabbitMQ vhost:

```
cp -f $MOOGSOFT_HOME/etc/cots/rabbitmq/rabbitmq.config
$MOOGSOFT_HOME/cots/rabbitmq-server/etc/rabbitmq/
bash $MOOGSOFT_HOME/bin/utils/moog_init_mooms.sh -z <VHOST> -p
```

Upgrade Apache Tomcat and Nginx

Enter the following commands to upgrade Apache Tomcat and Nginx:

```
$MOOGSOFT_HOME/bin/utils/moog_init_ui.sh -tnfwz
$( $MOOGSOFT_HOME/bin/utils/moog_config_reader -k mooms.zone) --accept-eula
```

Migrate the certificates for the previous deployment of Nginx to the new deployment. Use the following commands as an example of how to do this in the general case where SSL terminates in Nginx (default configuration):

```
cp -f $CERT_REAL_PATH_PEM $MOOGSOFT_HOME/dist/7.2.0/cots/nginx/ssl/
cp -f $CERT_REAL_PATH_KEY $MOOGSOFT_HOME/dist/7.2.0/cots/nginx/ssl/
sed -i "s|.*ssl_certificate.*|${CERT_PATH_PEM}|"
$MOOGSOFT_HOME/dist/7.2.0/cots/nginx/config/conf.d/moog-ssl.conf
sed -i "s|.*ssl_certificate_key.*|${CERT_PATH_KEY}|"
$MOOGSOFT_HOME/dist/7.2.0/cots/nginx/config/conf.d/moog-ssl.conf
```

Restart Nginx:

```
MOOGSOFT_HOME/bin/utils/process_cntl nginx restart
```

Update and Reconfigure Integrations

This upgrade process replaces the Moobot file \$MOOGSOFT_HOME/bots/moobots/RemedyIntegration.js Moobot file. If you have customised this file, back it up before continuing.

To update the integrations:



1. The LAMs/Integrations no longer use the 'state' files under \$MOOGSOFT_HOME/config. Remove these files with the following commands:

```
rm -f $MOOGSOFT_HOME/config/*_lam.state;
rm -f $MOOGSOFT_HOME/config/{null.state, TestStateManager.state}
rm -f $MOOGSOFT_HOME/config/observe*.conf
```

2. Run the following command to extract the old and new Integrations:

```
bash $MOOGSOFT_HOME/bin/utils/integration_installer -a -l WARN;
for file in $(diff -qr $MOOGSOFT_HOME/dist/7.2.0/ui/integrations/ \
$MOOGSOFT_HOME/dist/7.0.*ui/integrations/ | egrep -vi 'only
in.*7.2.0|differ' \
| awk '{print $NF}'); do cp -rp
$MOOGSOFT_HOME/dist/7.0.*ui/integrations/$file \
$MOOGSOFT_HOME/dist/7.2.0/ui/integrations/; done
```

Configure the ServiceNow MID Server to use Java 8

If you are using a ServiceNow MID server installed on the same host as Cisco Crosswork Situation Manager, you must configure it to point to Java 8. The MID server requires Java 8 (update 152 or later). It will not work with Java 9+. To do this:

1. Install the latest version of Java 8. See the [ServiceNow MID server system requirements](#) for more information.

2. Stop the MID server by running the appropriate command. For example:

```
kill -9 $(ps -ef | grep mid_server | grep -v grep | awk '{print $2}')
```

3. Configure the wrapper.java.command property to point to the Java 8 binary in the following file:

```
/usr/local/servicenow/moog_mid_server/agent/conf/wrapper-override.conf.
```

For example:

```
wrapper.java.command=/usr/java/jre1.8.0_171-amd64/bin/java
```

In the next step you will reconfigure the ServiceNow integration which will restart the MID server.

Reconfigure your UI Integrations

Delete and reconfigure all of your UI integrations, using the configuration details you noted prior to the upgrade. This is required due to UI changes in the new version.

Confirm All Processes are Running and Start Moogfarmd

To confirm all services are running:

1. Run the following commands to verify that all required processes are running and start Moogfarmd:



```
$MOOGSOFT_HOME/bin/utils/process_cntl apache-tomcat status
$MOOGSOFT_HOME/bin/utils/process_cntl moog_farmd start
$MOOGSOFT_HOME/bin/utils/process_cntl nginx status
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch status
```

2. Run the following command to re-enable the Event Analyser cronjobs:

```
(crontab -l | sed -e 's/^#\+\.events analyser.\+/\1/') | crontab -
```

- Run the following command to restart any previously running UI-based Integrations:

```
bash $MOOGSOFT_HOME/bin/utlis/startup cntl;
```

Add the Keepalive Cronjob (Optional)

A new cronjob is installed for clean installations of 7.2.0 which provides the ability for certain services to be restarted automatically should they fail. For RPM deployments this is usually handled by chkconfig or MySQL itself for example, but can be useful in certain cases. See [Configure Services to Restart](#) for more information.

Command to enable if required:

```
(crontab -l; echo -e "*\t*\t*\t*\t*\n$MOOGSOFT_HOME/bin/utils/process_keepalive.sh 2>&1") | crontab -
```

Migrate the Java Keystore

The upgrade includes a new version of the Java Runtime, so you need to migrate any certificates stored in the old Java keystore into the new keystore, which is now the JRE under `$MOOGSOFT_HOME/cots/jre/`.

The previous JRE is located in `$MOOGSOFT_HOME/dist/7.0.*/cots/jre`.

If you did not manually add certificates to the old store, you can skip this step.

Verify the Upgrade

You can verify the upgrade manually or using automatic utilities.

Verify the Upgrade Manually

Perform the following basic steps to ensure the upgrade to Cisco Crosswork Situation Manager v7.2.0 was successful:

1. Verify that the UI login page displays "Version 7.2.0" at the top.
2. Check that the UI "Support Information" window correctly indicates the current version as "7.2.0" and shows the correct schema upgrade history.

Verify the Upgrade Using Automatic Utilities

Run the following automatic utilities to ensure that the upgrade to 7.2.0 was successful:



1. Confirm that all Cisco Crosswork Situation Manager files have been deployed correctly within `$MOOGSOFT_HOME` using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_install_validator.sh
```

2. Confirm that all Apache Tomcat files have been deployed correctly within `$MOOGSOFT_HOME` using this utility:

```
$MOOGSOFT_HOME/bin/utils/tomcat_install_validator.sh
```

- a. Verify that all the steps have completed successfully. If there are some web app differences, run the following command to extract the web app with the correct files:

```
$MOOGSOFT_HOME/bin/utils/moog_init_ui.sh -w
```

3. Confirm that the database schema has upgraded successfully using this utility:

```
$MOOGSOFT_HOME/bin/utils/moog_db_validator.sh
```

Confirm that all the steps are successful. Some schema differences might be valid (e.g. `custom_inforelated`). If there are more substantial differences, you should investigate further to verify that all the pre-requisite upgrade scripts have been applied in the right order:

```
mysql -u root <moogdb_database_name> -e "select * from  
schema_upgrades;"  
mysql -u root <moog_reference_database_name> -e "select * from  
schema_upgrades;"
```

Remove the MySQL Connector

After upgrading to CCSM 7.2 go to all your machines and remove the file:

```
$MOOGSOFT_HOME/lib/cots/nonDist/mysql-connector-java-  
5.1.45.jar
```

Troubleshooting

If you have any issues, refer to [Troubleshooting](#).

Upgrade Tarball v7.1.x to Tarball v7.2.x

This topic describes the upgrade procedure of Tarball deployments from Cisco Crosswork Situation Manager v7.1.x Tarball to Cisco Crosswork Situation Manager v7.2.x Tarball.

Before You Begin

To perform the upgrade you must be logged in as the same non-root user that installed and runs the existing Cisco Crosswork Situation Manager v7.1.x software.



[Download the Installer](#)

Run the following command to download the installer:

```
curl -L -O  
"https://<username>:<password>@speedy.moogsoft.com/installer/moogsoft-  
aiops-7.2.0.tgz"
```

[Note UI Integrations Configuration](#)

If you are using any of the following UI integrations, make a note of their connection and configuration details before you begin the upgrade. You will need to delete and reconfigure them in the post-upgrade steps, due to UI changes in the new version.

- AWS CloudWatch
- Cherwell
- JIRA Service Desk
- JIRA Software
- JMS
- New Relic
- Remedy
- ServiceNow
- SevOne
- Slack
- SolarWinds
- VMware vCenter
- VMware vSphere
- vRealize Log Insight
- WebSphere MQ
- xMatters

[Stop Services and Processes](#)

1. Stop Moogfarmd:

```
$MOOGSOFT_HOME/bin/utils/process_cntl moog_farmd stop
```

2. Stop Apache Tomcat:

```
$MOOGSOFT_HOME/bin/utils/process_cntl apache-tomcat stop
```



3. Stop the integrations and LAMs, using either the Process Control utility or the kill command:

```
$MOOGSOFT_HOME/bin/utils/process_cntl <lam_name> stop
```

or:

```
kill -9 $(ps -ef | grep java | grep _lam | awk '{print $2}')  
2>/dev/null
```

4. Disable the Events Analyser from running during the upgrade process:

a. Run the following command to comment out the relevant lines in crontab:

```
(crontab -l | sed -e 's/^\(.*events_analyser.*\)$/#\1/' ) | crontab -
```

b. Run the following command to stop any active Events Analyser processes:

```
ps -ef | grep java | egrep 'events_analyser|moog_indexer' | awk '{print  
$2}' | xargs kill 2>/dev/null
```

5. Stop MySQL, RabbitMQ, Elasticsearch, Nginx and Apache Tomcat:

```
$MOOGSOFT_HOME/bin/utils/process_cntl mysqld stop  
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch stop  
$MOOGSOFT_HOME/bin/utils/process_cntl rabbitmq stop  
$MOOGSOFT_HOME/bin/utils/process_cntl nginx stop
```

6. Obtain the path variables before re-linking the directories. From this step onwards, use the same terminal session to keep the variables:

```
CERT_REAL_PATH_PEM=$(readlink -f $(grep -h 'ssl_certificate '  
$MOOGSOFT_HOME/dist/7.1.*/cots/nginx/config/conf.d/moog-ssl.conf|head -  
1|awk '{print $2}'|tr -d ';'))  
CERT_REAL_PATH_KEY=$(readlink -f $(grep -h 'ssl_certificate_key'  
$MOOGSOFT_HOME/dist/7.1.*/cots/nginx/config/conf.d/moog-ssl.conf|head -  
1|awk '{print $2}'|tr -d ';'))  
CERT_PATH_PEM=$(grep -h 'ssl_certificate '  
$MOOGSOFT_HOME/dist/7.1.*/cots/nginx/config/conf.d/moog-ssl.conf|head -  
1)  
CERT_PATH_KEY=$(grep -h 'ssl_certificate_key'  
$MOOGSOFT_HOME/dist/7.1.*/cots/nginx/config/conf.d/moog-ssl.conf|head -  
1)
```

Back Up the Existing System

To back up the existing system:

1. Back up \$MOOGSOFT_HOME.
2. Take a snapshot (for VMs).
3. Back up MySQL.



Upgrade Cisco Crosswork Situation Manager

Run the following commands to perform the upgrade:

```
tar -xf moogsoft-aiops-7.2.0.tgz
bash moogsoft-aiops-install-7.2.0.sh
```

Follow the instructions that appear. The upgrade process detects the v7.1.0 installation and performs an upgrade (a side-by-side deployment).

Run the following command to change the default log level for Elasticsearch. By default it is set to INFO but you can change it to WARN to reduce the size of the Elasticsearch logs.

```
sed -i -e "s;rootLogger.level = info;rootLogger.level = warn;g"
$MOOGSOFT_HOME/cots/elasticsearch/config/log4j2.properties
```

Run the following commands to restart the MySQL, RabbitMQ and Elasticsearch processes:

```
$MOOGSOFT_HOME/bin/utils/process_cntl mysqld start
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch start
$MOOGSOFT_HOME/bin/utils/process_cntl rabbitmq start
```

Merge the Latest Configuration File Changes

The top-level `$MOOGSOFT_HOME/config` and `$MOOGSOFT_HOME/bots` folders are the master folder locations for configuration and bot files. The new 'default' v7.2.0 versions of these files are stored in `$MOOGSOFT_HOME/dist/7.2.0/config/` and `$MOOGSOFT_HOME/dist/7.2.0/bots/` respectively.

1. Identify the config files that have changed between v7.1.0 and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/7.1.*/config
$MOOGSOFT_HOME/dist/7.2.0/config | grep -i 'differ'
```

2. For the files highlighted by the command above, the differences and new properties in the 7.2.0 versions of these files need to be manually merged into the files in `$MOOGSOFT_HOME/config`. Do not overwrite the files in `$MOOGSOFT_HOME/config`.

3. Identify the contrib files that have changed between v7.1.0 and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/7.1.*/contrib
$MOOGSOFT_HOME/dist/7.2.0/contrib | grep -i 'differ'
```

4. For the files highlighted by the command above, the differences and new properties in the 7.2.0 versions of these files need to be manually merged into the files in `$MOOGSOFT_HOME/contrib`. Do not overwrite the files in `$MOOGSOFT_HOME/contrib`.

5. Identify the bot files that have changed between v7.1.0 and v7.2.0. For example:

```
diff -rq $MOOGSOFT_HOME/dist/7.1.*/bots $MOOGSOFT_HOME/dist/7.2.0/bots
| grep -i 'differ'
```



6. For the files highlighted by the command above, the differences and new properties in the 7.2.0 versions of these files need to be manually merged into the files in `$MOOGSOFT_HOME/bots`. Do not overwrite the files in `$MOOGSOFT_HOME/bots`.

Upgrade the Cisco Crosswork Situation Manager Database Schema

To upgrade the Cisco Crosswork Situation Manager database schema, you must provide the Auto Upgrader utility with the credentials of a database user with super privileges. For single-host installations where MySQL was installed as part of the Cisco Crosswork Situation Manager deployment, you can use the default 'root' user.

1. Run the following command after substituting the `<MySQL-SuperUsername>` argument:

```
bash $MOOGSOFT_HOME/bin/utils/moog_db_auto_upgrader -t 7.2.0 -u <MySQL-SuperUsername>
```

2. Enter the password for that user. You can provide the password to the utility with the `-p` flag but this is not recommended in non-test deployments for security reasons.

Upgrade Apache Tomcat and Nginx

Enter the following commands to upgrade Apache Tomcat and Nginx:

```
$MOOGSOFT_HOME/bin/utils/moog_init_ui.sh -tnfwz  
$( $MOOGSOFT_HOME/bin/utils/moog_config_reader -k mooms.zone ) --accept-eula
```

Migrate the certificates for the previous deployment of Nginx to the new deployment. Use the following commands as an example of how to do this in the general case where SSL terminates in Nginx (default configuration):

```
cp -f $CERT_REAL_PATH_PEM $MOOGSOFT_HOME/dist/7.2.0/cots/nginx/ssl/  
cp -f $CERT_REAL_PATH_KEY $MOOGSOFT_HOME/dist/7.2.0/cots/nginx/ssl/  
sed -i "s|.*ssl_certificate.*|${CERT_PATH_PEM}|"   
$MOOGSOFT_HOME/dist/7.2.0/cots/nginx/config/conf.d/moog-ssl.conf  
sed -i "s|.*ssl_certificate_key.*|${CERT_PATH_KEY}|"   
$MOOGSOFT_HOME/dist/7.2.0/cots/nginx/config/conf.d/moog-ssl.conf
```

Restart Nginx:

```
$MOOGSOFT_HOME/bin/utils/process_cntl nginx restart
```

Update and Reconfigure Integrations

This upgrade process replaces the `$MOOGSOFT_HOME/bots/moobots/RemedyIntegration.js` Moobot file. Back up this file if you have customised it, before continuing.

To update the integrations:

1. The LAMs/Integrations no longer use the 'state' files under `$MOOGSOFT_HOME/config`. These files can be removed with the following command:



```
rm -f $MOOGSOFT_HOME/config/*_lam.state;
rm -f $MOOGSOFT_HOME/config/{null.state, TestStateManager.state}
rm -f $MOOGSOFT_HOME/config/observe*.conf
```

2. Run the following command to extract the old and new Integrations:

```
bash $MOOGSOFT_HOME/bin/utils/integration_installer -a -l WARN;
for file in $(diff -qr $MOOGSOFT_HOME/dist/7.2.0/ui/integrations/ \
$MOOGSOFT_HOME/dist/7.1.*ui/integrations/ | egrep -vi 'only
in.*7.2.0|differ' \
| awk '{print $NF}'); do cp -rp
$MOOGSOFT_HOME/dist/7.1.*ui/integrations/$file \
$MOOGSOFT_HOME/dist/7.2.0/ui/integrations/; done
```

Configure the ServiceNow MID Server to use Java 8

If you are using a ServiceNow MID server installed on the same host as Cisco Crosswork Situation Manager, you must configure it to point to Java 8. The MID server requires Java 8 (update 152 or later). It will not work with Java 9+. To do this:

1. Install the latest version of Java 8. See the [ServiceNow MID server system requirements](#) for more information.
2. Stop the MID server with the appropriate command. For example:

```
kill -9 $(ps -ef | grep mid_server | grep -v grep | awk '{print $2}')
```

3. Configure the `wrapper.java.command` property to point to the Java 8 binary in the following file:

```
/usr/local/servicenow/moog_mid_server/agent/conf/wrapper-override.conf.
```

For example:

```
wrapper.java.command=/usr/java/jre1.8.0_171-amd64/bin/java
```

In the next step you will reconfigure the ServiceNow integration which will restart the MID server.

Reconfigure your UI Integrations

Delete and reconfigure all of the following UI integrations, using the configuration details you noted prior to the upgrade. This is required due to UI changes in the new version.

- AWS CloudWatch
- Cherwell
- JIRA Service Desk
- JIRA Software



- JMS
- New Relic
- Remedy
- ServiceNow
- SevOne
- Slack
- SolarWinds
- VMware vCenter
- VMware vSphere
- vRealize Log Insight
- WebSphere MQ
- xMatters

Confirm All Processes are Running and Start Moogfarmd

To confirm all services are running:

1. Run the following commands to verify that all required processes are running and start Moogfarmd:

```
$MOOGSOFT_HOME/bin/utils/process_cntl apache-tomcat status
$MOOGSOFT_HOME/bin/utils/process_cntl moog_farmd start
$MOOGSOFT_HOME/bin/utils/process_cntl nginx status
$MOOGSOFT_HOME/bin/utils/process_cntl elasticsearch status
```

2. Run the following command to re-enable the Event Analyser cronjobs:

```
(crontab -l | sed -e 's/^#\+\\(.*events_analyser.*\\)/\\1/' ) | crontab -
```

3. Run the following command to restart any previously running UI-based Integrations:

```
bash $MOOGSOFT_HOME/bin/utils/startup_cntl;
```

Add the Keepalive Cronjob (Optional)

A new cronjob is installed for clean installations of 7.2.0 which provides the ability for certain services to be restarted automatically should they fail. For RPM deployments this is usually handled by chkconfig or MySQL itself for example, but can be useful in certain cases. See [Configure Services to Restart](#) for more information.

Command to enable if required:

Confirm that all the steps are successful. Some schema differences might be valid (e.g. `custom_inforelated`). If there are more substantial differences, you should



investigate further to verify that all the prerequisite upgrade scripts have been applied in the right order:

```
mysql -u root <moogdb_database_name> -e "select * from  
schema_upgrades;"  
mysql -u root <moog_reference_database_name> -e "select * from  
schema_upgrades;"
```

Remove the MySQL Connector

After upgrading to CCSM 7.2 go to all your machines and remove the file:

```
$MOOGSOFT_HOME/lib/cots/nonDist/mysql-connector-java-  
5.1.45.jar
```

Troubleshooting

If you have any issues, refer to [Troubleshooting](#).

Post-installation Setup

Shortly after you install the Cisco Crosswork Situation Manager packages and the system is running, you can configure additional components per your organizations needs:

- [Apply Valid SSL Certificates](#)
- [Configure External Authentication](#)
- [Configure Search and Indexing](#)
- [Configure Services to Restart](#)
- [Configure the Tool Runner](#)
- [Configure SMS Notifications](#)
- [Enable Situation Room Plugins](#)
- [Import a Topology](#)

If you need to troubleshoot Cisco Crosswork Situation Manager:

- [Monitor and Troubleshoot Moogsoft AIOps](#)
- [Configure Logging](#)

After your system has been running for some time:

- [Configure Historic Data Retention](#)
- [Archive Situations and Alerts](#)

Apply 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 nginx config folder and edit `/etc/nginx/conf.d/moog-ssl.conf`:

```
vi /etc/nginx/conf.d/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
```

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="}
```

Note

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" : "e5u00LY3HQJZCltG/caUnVbxVN4hImm4gIOpb4rwpF4=",
```

3. Set the value of the `encrypted_password` property to the value returned from `themoog_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"
```



}
#

Note

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.

Configure External Authentication

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

- [Configure Single Sign-On with LDAP](#)
- [Configure Single Sign-On with SAML](#)
- [Security Configuration Reference](#)

Configure Single Sign-On with SAML

You can configure Cisco Crosswork Situation Manager so users from an external directory can log in by Single Sign-On (SSO) using Security Assertion Markup Language (SAML).

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's 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 webhost URL in `$MOOGSOFT_HOME/config/servlets.conf` is the same as your Cisco Crosswork Situation Manager instance URL:

```
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/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. Configure the roles, teams and primary group mappings for new users that log in to Cisco Crosswork Situation Manager using SAML. These are all required:
 - **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. These are all required:
 - **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 additional IdP attribute mappings:
- **contactNumber:** Defines the IdP attribute that maps to contact number in Cisco Crosswork Situation Manager.
 - **department:** Defines the IdP attribute that maps to department in Cisco Crosswork Situation Manager.
 - **primaryGroup:** Defines the IdP attribute that maps to primary group in Cisco Crosswork Situation Manager.
 - **timezone:** Defines the IdP attribute that maps to timezone in Cisco Crosswork Situation Manager.
 - **teamAttribute:** Defines the IdP attribute that maps to teams in Cisco Crosswork Situation Manager.
 - **teamMap:** Defines the IdP attribute or custom attribute that maps to team names in Cisco Crosswork Situation Manager.
 - **createNewTeams:** Creates a team or teams if they did not exist in Cisco Crosswork Situation Manager.
 - **roleAttribute:** Defines the IdP attribute containing role information.
 - **roleMap:** Defines the IdP attribute that maps to Cisco Crosswork Situation Manager roles.
7. 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.
8. 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.
9. Optionally configure the Service Provider Entity Id. See [Optional SAML Security Features](#):
- **serviceProviderEntityId:** Service Provider Entity ID assertion number.



10. 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.

```
"maximumAuthenticationLifetime": 3600
```

Enable Entity ID Assertion

You can enable 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:

```
"serviceProviderEntityId": "MoogsoftAI0ps"
```

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`, `defaultTeams` and `defaultGroup` properties in the SAML realm configuration.

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

```
"assignTeams": {  
  "teamAttribute": "groups",  
  "teamMap": {  
    "IdP Team": "Moogsoft AIOps Team",  
    "Another IdP Team": "Another AIOps team"  
  }  
}
```

To create a team that does not exist already, enable the `createNewTeams` property:

```
"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": {  
  "roleAttribute": "groups",  
  "roleMap": {  
    "IdP Standard User" : "Operator",  
    "IdP Manager User"  : "Manager"  
  }  
}
```



Note

You must map both roles and teams through IdP to prevent users being assigned to the default role and team.

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",
  "createNewTeams": true,
  "teamMap": {
    "Cloud Team": "Cloud DevOps",
    "Database Team": "Database DevOps"
  }
},
"assignRoles" : {
  "roleAttribute": "groups",
  "roleMap" : {
    "Standard User": "Operator",
    "Manager User": "Manager"
  }
},
"keystorePassword": "my_realm_secret",
"privateKeyPassword": "my_realm_secret",
"maximumAuthenticationLifetime": 60,
"serviceProviderEntityId": "MoogsoftAI0ps"
}
```

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/jCCAeagAwIBAgIQCGehfcnv6r5My/fnrbfDejANBgkqhkiG9w0BAQsFADAV
MRMwEQYDVQQDEwp3d3cuc3AuY29tMB4XDTEzMTEyMjA4MjMyMVoXDTQ5MTIzMTE0
MDAwMFowFTETMBEGA1UEAxMKd3d3LnNwLmNvbTCCASIwDQYJKoZIhvcNAQEBBQAD
ggEPADCCAQoCggEBAMPm/ew9jaGWPQS1C7KtpvgzV4nSOIFPgRt/nlRYR+pUWdDE
```



```
fSKmyjK28nkQ1KKujRJTnvmZydmUrmEFpVv+giBiUkvCJY3PxZ/EDSsF3R/OzWh
kUv5nfAXPnqkX9x22b6+vUof6WiLGyAW6lOYMCVADjTS19pSaUtIaANDx9maERcT
9eQbGSnjim0WurFRYs9ZE8ttErrMH9+Su4246YDqOPAkz6La4cHHMPQdcFQT5p/c
uXBfU1v11tWdBEGAY3xHYZE8u5TTJ/vp9UxyU1Mwfe02g9VDRcokLQHrj6wFxtvu
fA+WtUKYJGUu2p/qSuaw7eS6UFjUn49aVqg9OacCAwEAAaANKMEgwRgYDVR0BBBD8w
PYAQ1/S0ibdvfdFkJ9T9oIPluKEXMBUxEzARBgNVBAMTCnd3dy5zcC5jb22CEAhn
oX3J7+q+TMv35623w3owDQYJKoZIhvcNAQELBQADggEBAAHlmVoAZUt6paeFvtQb
c/iaJe/Fhd+JG1U0jyj1FDcCn8erLihEbhb3mFBBMF25o067gfA1JJXZrmHry3N1
OZuovqRqm8v7wg8n0nQa1HUWkUC2TBgfg1HE8/2rmSF2PngiEi18VOxRDxx0WXMN
ZX6JebJ1kCOCpT/x7aupS7T1GrIPmDLxjnC9Bet7pRynfomjP/6iU21/xOIF6xB9
Yf1a/kQbYdAVt2haYKI fvaF3xsq1X5tCXc9ijhBMgyaoqA+bQJD/13S8+yCmMxEY
ZjAVLEkyG1U4Uwo01cKEYbXIG/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: MIIDijCCAnICCQD[...]+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"

Configure Single Sign-On with LDAP

You can configure Cisco Crosswork Situation Manager so users from an external directory can log in by [Single Sign-On](#) (SSO) using [Lightweight Directory Access Protocol](#) (LDAP).

See [LDAP version 3](#) for more information.

Before You Begin

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

- You have the URL for your LDAP server.
- If you want to use a "lookup" DN ([Distinguished Name](#)) resolution method, you have the credentials for the LDAP user who has rights to look up other users and determine their roles.
- If you want to use SSL encryption, you have a valid SSL certificate.

Configure LDAP for Cisco Crosswork Situation Manager

Edit the configuration file to configure and enable LDAP for Cisco Crosswork Situation Manager. You can find the file at \$MOOGSOFT_HOME/config/security.conf.



See the [Security Configuration Reference](#) for a full description of all properties. Some properties in the file are commented out by default. Uncomment properties to enable them.

1. Configure the properties for the LDAP connection:
 - **url**: URL of your LDAP server. This is required.
 - **connectionTimeout**: Connection timeout in milliseconds.
 - **readTimeout**: Read timeout in milliseconds.
 - **predefinedUser**: Determines if user must exist in the local database or not.
2. Configure the user resolution and attribute search section:
 - **resolutionType**: Type of DN resolution method. Valid options are "direct" and "lookup".
 - **attributeSearchFilter**: Defines an optional attribute filter to retrieve all user attributes.
 - **attributeMap**: Defines an attribute map between the LDAP user attributes and the user attributes in the Cisco Crosswork Situation Manager database.
3. Configure the LDAP group search section:
 - **systemUser**: Username of the system user to bind and search for user group information.
 - **systemPassword**: Password of the system user to bind and search for user group information.
 - **groupBaseDn**: Defines a group base DN to search for LDAP groups.
 - **memberAttribute**: Attribute used look for group members. Defaults to "member".
 - **groupNameAttribute**: Attribute used to look for group name.
 - **roleMap**: Defines the role mappings between the user directory and Cisco Crosswork Situation Manager.
 - **assignTeams**: Synchronizes team assignment between the user directory and the teams in Cisco Crosswork Situation Manager.
4. Optionally configure SSL if you want to enable TLS authentication:
 - **ssl_protocol**: Defines the SSL protocol you want to use. Defaults to TLSv1.2.
 - **server_cert_file**: SSL server certificate.
 - **client_cert_file**: Client certificate file.



- **client_key_file**: Client key file.

When you have finished your configuration, restart Apache Tomcat to activate any changes you made to the configuration file:

```
service apache-tomcat restart
```

See [Control Moogsoft AIOps Processes](#) for further details.

Configuration Example

An example of an LDAP configuration that uses direct DN resolution and SSL without client authentication:

```
"Example_Ldap" : {
  "realmType": "LDAP",
  "url": "ldap://moogsaml:389",
  "userDnResolution":
  {
    "resolutionType" : "direct",
    "direct" : {
      "usernameAttribute": "uid",
      "userDnPostfix": "ou=People,dc=moogsoft,dc=com"
    }
  },
  "attributeMap": {
    "fullname": "cn",
    "email": "mail"
  },
  "groupBaseDn": "ou=Group,dc=moogsoft,dc=com",
  "memberAttribute": "member",
  "groupNameAttribute": "cn",
  "roleMap": {
    "role-admin": "Super User",
    "OperatorRole": "Operator"
  },
  assignTeams :{
    teamMap: {
      CloudDevOps: "Cloud DevOps team",
      DBDevOps: "Database DevOps team"
    },
    useGroupName : true,
    createNewTeams: true
  }
  , "ssl":
  {
    "server_cert_file" : "/usr/share/moogsoft/config/example.crt"
  }
},
```



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 [LDAP](#) and [SAML](#).

LDAP Connection Properties

You can configure the LDAP connection using the following properties:

url

The protocol (LDAP or LDAPS) along with the host and port of your LDAP server. For example: `ldap://172.16.124.169:389`.

Type: String

Required: Yes

Default: N/A

connectionTimeout

Defines the connection timeout in milliseconds.

Type: String

Required: Yes

Default: 30000

readTimeout

Defines the read timeout in milliseconds.

Type: String

Required: Yes

Default: 30000

predefinedUser

If enabled, the user account information must exist in the local database as well as the LDAP server and predefined user details are used to populate created or updated user accounts.

If disabled, Cisco Crosswork Situation Manager creates or updates user accounts with the LDAP information.

Type: String

Required: Yes

Default: False



LDAP Attribute Search Properties

You can configure the authentication bind, DN resolution method and attribute search with the following properties:

resolutionType

Defines the method to look up the DN ([Distinguished Name](#)), a unique path to any object in the active directory.

Type: String

Required: Yes

One of: direct, lookup

Default: N/A

There are two methods to choose from:

- **direct:** If using this method, the user DN is created using the `usernameAttribute` and `userDnPostfix` properties. These properties are required. For example:

```
"userDnResolution": {
  "resolutionType" : "direct",
  "direct" : {
    "usernameAttribute": "uid",
    "userDnPostfix": "ou=People,dc=moogsoft,dc=com"
  }
},
```

For a user called John Smith, the user DN is:

```
uid=john.smith,ou=People,dc=moogsoft,dc=com
```

- **lookup:** If using this method, Cisco Crosswork Situation Manager searches for the user in the LDAP server using a combination of `usernameAttribute` and `userBaseSearchFilter` as a filter and `userBaseDn` as a base to find the DN. These properties are required. For example:

```
"userDnResolution": {
  "resolutionType" : "lookup",
  "lookup" : {
    "usernameAttribute": "sAMAccountName",
    "userBaseDn" : "ou=People,dc=moogsoft,dc=com",
    "userBaseSearchFilter" : "(objectclass=person)",
  }
},
```

Optionally for both "direct" and "lookup" methods, you can use the `userDnLookupUser`, `userDnLookupPassword` and `encryptedUserDnLookupPassword` properties to define the user to look up each DN



in your directory. See [Moog Encryptor](#) for more information if you want to use password encryption.

If you leave the `userDnLookupUser` property empty, LDAP uses the `systemUser` defined in the LDAP Group Search section instead.

attributeSearchFilter

Defines an optional LDAP attribute filter to search for user attributes.

Type: String

Required: No

Default: (`objectclass=*`)

attributeMap

Defines an attribute map between the LDAP user attributes and the user attributes in the Cisco Crosswork Situation Manager database.

Type: String

Required: No

Default: N/A

This property uses the format:

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

LDAP Group Search and Mapping

You can configure the following properties in the LDAP group search section:

systemUser

Username of the system user to bind and search for user group information. LDAP uses this user if you leave the `userDnLookupUser` property empty. The system sends two bind requests and two search requests with LDAP. If you do not configure a system user, the user bind chosen for authentication is also used for the LDAP group search.

Type: String

Required: No

Default: N/A

systemPassword

Password of the system user to bind and search for user group information.



Type: String

Required: No

Default: N/A

groupBaseDn

DN for the part of the LDAP structure that contains the user groups. This is used in conjunction with the `memberAttribute` to find any LDAP groups the user belongs to. These groups are then mapped to a local role using the `roleMap` property.

Type: String

Required: No

Default: N/A

memberAttribute

Attribute used to look for group members.

Type: String

Required: No

Default: member

groupNameAttribute

Attribute used to look for group name.

Type: String

Required: No

Default: CN

roleMap

Defines the role mappings between the user directory and Cisco Crosswork Situation Manager.

Type: String

Required: No

Default: N/A

LDAP AssignTeams Properties

You can configure the following sub-properties of `assignTeams` to synchronize team assignment between the user directory and the teams in Cisco Crosswork Situation Manager.

assignTeams



Synchronizes team assignment between the user directory and the teams in Cisco Crosswork Situation Manager.

Type: String

Required: No

Default: N/A

teamMap

Defines the LDAP attribute or custom attribute that maps to team names in Cisco Crosswork Situation Manager. You can provide the mapping as a JSON object. For example:

```
{ "LDAP Team" : "Moogsoft Team", "Another LDAP Team" : "Another Moogsoft team" }
```

Type: JSON Object

Required: No

Default: N/A

useGroupName

Enable to use the LDAP group name as the team name in Cisco Crosswork Situation Manager.

Type: Boolean

Required: No

Default: false

createNewTeams

Creates a team or teams if they do not exist in Cisco Crosswork Situation Manager. If you leave teamMap empty, the teams adopt their LDAP teams names.

Type: Boolean

Required: No

Default: false

LDAP SSL Properties

You can optionally configure SSL to enable TLS authentication:

ssl_protocol

Defines the SSL protocol you want to use.

Type: String



Required: No

Default: TLSv1.2

server_cert_file

SSL server certificate.

Type: String

Required: No

Default: N/A

client_cert_file

SSL client certificate.

Type: String

Required: No

Default: N/A

client_key_file

Client key file.

Type: String

Required: No

Default: N/A

SAML Service Provider Properties

You can configure a SAML realm by giving it a name and editing 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 must 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: Yes

Default: "\$Email"

email

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

Type: String

Required: Yes

Default: "\$Email"

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

Type: String

Required: Yes

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 timezone in Cisco Crosswork Situation Manager.

Type: String

Required: No

Default: "timezone",

SAML assignTeams Properties

You can configure the following sub-properties of `assignTeams` to synchronize team assignment between the SAML user directory and the teams in Cisco Crosswork Situation Manager:

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" : "Moogsoft AIOps Team", "Another IdP Team" : "Another AIOps team" }

createNewTeams

Creates a team or teams if they do not exist in Cisco Crosswork Situation Manager. If you leave teamMap empty, the teams adopt their IdP teams names.

Type: Boolean

Required: No

Default: false

SAML assignRoles Properties

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" }

SAML Security Properties

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 Message Bus](#)

The Moogsoft 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 are lost. Durable queues can be enabled using the `message_persistence` setting in `$MOOGSOFT_HOME/config/system.conf` (see [Message Persistence](#)).

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 components (LAMs, Moogfarmd, the Moolets, etc.).

The out-of-the-box configuration in `$MOOGSOFT_HOME/config/system.conf` is as follows:

```
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 `$MOOGSOFT_HOME/config/system.conf`:

message_persistence

Controls whether the Message Bus persists important messages 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 resend 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 is 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 in the cache list. The system attempts to resend any cached messages in the order in which they were put on the cache until their cache time-to-live (`cache_ttl`) value has been reached. Any messages not sent successfully sent is discarded. This value has 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.

Warning

Cisco 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, the Cisco Crosswork Situation Manager installation includes with a single RabbitMQ broker running on the same server as the other Cisco Crosswork Situation Manager components (LAMs, moogfarmd, the Moolets, etc.). You can also configure Cisco Crosswork Situation Manager as a distributed system with multiple RabbitMQ broker hosts.

If you encounter any errors or issues 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 package built by RabbitMQ:

```
yum -y install https://github.com/rabbitmq/erlang-  
rpm/releases/download/v20.1.7/erlang-20.1.7-1.el7.centos.x86_64.rpm
```

2. Set up RabbitMQ yum repository:

```
curl -s https://packagecloud.io/install/repositories/rabbitmq/rabbitmq-  
server/script.rpm.sh | sudo bash
```

3. Install RabbitMQ:

```
yum -y install rabbitmq-server-3.7.4
```

4. 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
```

5. Run the following commands:

```
chkconfig rabbitmq-server on  
service rabbitmq-server start
```

6. 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>" ...
```

7. Edit the "mooms" section in `system.conf` on the Cisco 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](#).

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, if the zone is set to Cisco Crosswork Situation Manager:

```
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 (sometimes called 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 Message Bus 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 `socketlamd` 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 installation documentation](#).

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': eaccess",[]}
```

```
{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': eaccess", [{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 Message Bus 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(Autore
coveringChannel.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.jav
a:378)
at com.moogsoft.mooms.CJNIMoomsWrapper.sendEvent(CJNIMoomsWrapper.java:288)
|+
```



Note

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 initd files 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:



1. Configure the contents of `/etc/rabbitmq/rabbitmq-env.conf`, for example:

```
RABBITMQ_NODE_IP_ADDRESS="172.168.87.131"
RABBITMQ_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.conf` match 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.conf` match those defined in the RabbitMQ broker with the above commands.

Message System SSL

The Message Bus system (MooMs) can be configured to operate using SSL connections to provide secure and authorized connectivity.

The message system for Cisco Crosswork Situation Manager is 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 the [Rabbit MQ documentation](#).

Configure Cisco Crosswork Situation Manager to use SSL with the Message Bus

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 Message Bus configuration 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.

Note

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"
}
```

Note

To disable SSL connectivity with the Message Bus, change the port number for the brokers back to the non-SSL port (typically 5672) and comment out the "ssl" section in system.conf.

[Configure 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/
```

[Index Alerts and Situations](#)

Two tools are used 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 modified so that they are immediately searchable.



You can configure the Indexer in `$MOOGSOFT_HOME/config/moolets/indexer.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 the indexer applies team permissions to the indexes.Manage Teams

If disabled, the 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. This is useful because it is not possible to load all alerts to the memory at once.

By default the Indexer scans through batches of one thousand alerts or Situations.

Type: Integer

Default: 1000

full_scan_wait

The number of seconds the Indexer waits between batches. This frees up the CPU and memory used to index each batch.

It is set to zero 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 reindex. 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, the 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, the Indexer scans both databases every three days.

Type: Integer

Default: 3

By default the Indexer is configured as follows:

```
# 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

Before you can run the indexer utility, you must start Moogfarmd with a running Indexer Moolet. The `moog_indexer` 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	WARN INFO DEBUG TRACE	Specify the log level to choose the amount of debug output. Defaults to INFO.



<arg>

<code>-n, --now</code>	-	Schedules a full index to run immediately.
<code>-r, --report</code>	-	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.

Note

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
```

Warning

Before you upgrade to Cisco Crosswork Situation Manager V6.2.1 or later, 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.



Configure Logging

- [Configure Your Log Files](#)
- [Log Files by Component](#)
 - [Apache Tomcat](#)
 - [Nginx](#)
 - [Moogfarmd](#)
 - [LAMs and Integrations](#)
 - [MySQL](#)
 - [RabbitMQ](#)
 - [Elasticsearch](#)

- [Log Rotation](#)

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 the [Log Levels Reference](#) for details.

Cisco Crosswork Situation Manager uses Apache Log4j for logging. See the [Log4j configuration](#) documentation for more information.

Configure Your Log Files

You can edit the log configuration files at `$MOOGSOFT_HOME/config/logging/`

There is a configuration file for every component or servlet in Cisco Crosswork Situation Manager. These files can be found in `$MOOGSOFT_HOME/config/logging/servlets/` and follow the naming convention `<servlet_name>.log.json`. These configuration files control the logs for the following:

- `events.log.json`: Logs for the proxy LAM.
- `graze.log.json`: Graze request logs.
- `moogpoller.log.json`: Moogpoller logs.
- `moogsvr.log.json`: Logs relating to SAML/LDAP authentication and internal API calls.
- `situation_similarity.log.json`: Situation Similarity servlet logs.
- `toolrunner.log.json`: Toolrunner servlet logs.

The other default configuration files include:



- `moog_farmd.log.json`: Configures logs for Moogfarmd process.
- `moogsoft.log.json`: Configures logs for all of the utilities.
- `integrations.log.json`: Configures logs for LAMs and integrations.

You can change log levels and make other configuration changes to components while they are running. Cisco Crosswork Situation Manager reads any changes and applies them every two seconds.

You can configure these files to meet your requirements. Refer to the [Log4j documentation](#) to see the available properties or see [Log Configuration File Examples](#).

Log Files by Component

The following reference provides 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, edit the relevant files in `$MOOGSOFT_HOME/config/logging/servlets`.

Nginx

Log location: `/var/log/nginx`

Primary log file: `error.log`

To change the logging level for Nginx:

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

`LogLevel debug`

3. Restart Nginx.

Moogfarmd

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

MOO is the default HA cluster name in `$MOOGSOFT_HOME/config/system.conf`. If you change it the Moogfarmd log file path changes accordingly.

Restart Moogfarmd after making any of the following configuration changes.



To use a custom log configuration file for Moogfarmd:

1. Make a copy of the default Moogfarmd log configuration file and rename it, for example:

```
cd $MOOGSOFT_HOME/config/logging
cp moog_farmd.log.json mymoog_farmd.log.json
```

2. Edit the new file according to your Moogfarmd logging requirements.
3. Edit the `configuration_file` property in the `log_config` section of `moog_farmd.conf` to point to the new file. For example:

```
log_config:
{
    configuration_file: "mymoogfarmd.log.json"
}
```

To change the logging level for Moogfarmd:

1. Edit `/etc/init.d/moogfarmd`.
2. Set the `LogLevel` property. For available log levels see [Log Levels Reference](#). For example to enable logging at the debug level:

```
LOG_LEVEL=DEBUG
```

Alternatively, edit the level in `$MOOGSOFT_HOME/config/logging/moog_farmd.log.json`. For example:

```
"configuration":
{
    "ThresholdFilter":
    {
        "level": "trace"
    },
}
```

To save Moogfarmd logs to a different location and/or filename, edit the Moogfarmd log configuration file located at `$MOOGSOFT_HOME/config/logging/moog_farmd.log.json`.

For example:

```
"RollingFile":
{
    "name" : "FILE",
    "fileName" : "/var/log/moogsoft/Moogfarmd_test.log"
}
```

LAMs and Integrations

LAMs and integrations log their processing and data ingestion to two types of log files, process and capture.



Process Logs

LAMs and integrations record their activities as they ingest raw data. By default these process logs are written to a log file stored in `/var/log/moogsoft` if the user running the LAM has write permissions for this directory. Otherwise, the logs are written to `$MOOGSOFT_HOME/log`. By default the log file takes the name of the LAM or integration. For example, `MOO.solarwinds_lam.log`.

The configuration of LAM process logs is specified in a file located at `$MOOGSOFT_HOME/config/logging/integrations.log.json`.

To specify the log configuration for a particular LAM:

1. Make a copy of the default LAM log configuration file and rename it with the name of the LAM, for example:

```
cd $MOOGSOFT_HOME/config/logging
cp integrations.log.json solarwinds_lam.log.json
```

2. Edit the file according to your LAM logging requirements.
3. Edit the `configuration_file` property in the `log_config` section of the LAM configuration file to point to the new file. For example:

```
log_config:
{
    configuration_file:
"$MOOGSOFT_HOME/config/logging/solarwinds_lam.log.json"
}
```

If a polling integration or LAM fails to connect to the target system using the connection details in the UI or configuration file, Cisco Crosswork Situation Manager creates an alert with critical severity and writes the details to the process log. The following example shows a log file entry for a failed Zabbix Polling integration with an invalid URL:

```
WARN : [target1][20190117 13:03:33.942 +0000] [CZabbixPollingTask.java:129]
+|40001: An error response received
from Zabbix REST server: [Invalid URL provided
[http://zabbixserver1/zabbix/api_jsonrpc.php] for User Login request]|+
```

The following error code raises a Cisco Crosswork Situation Manager alert. The alert details are listed below:

External ID	Type	Class	Severity	Example Alert Description
40001	Internal Integrations Error	Failed Connection Attempt	Critical	Failed Connection Attempt for target [target1] and destination [http://zabbixserver1/zabbix/api_jsonrpc.php]. This is attempt [1] out of [infinite].



If the integration or LAM polls successfully on the next attempt, the alert is cleared. If the integration or LAM is restarted to resolve the connection issue the alert is not cleared and must be handled manually.

Capture Logs

In addition to process logs, all LAMs except the Logfile LAM allow you to capture the raw data they receive. This feature is disabled by default. To enable it, edit the LAM's configuration file and uncomment the `capture_log` property in the agent section. The default path to the capture log files is `$MOOGSOFT_HOME/log/data-capture/<lam_name>.log`.

An example agent section in a LAM configuration file is as follows:

```
agent:
{
    name          : "SolarWinds",
    capture_log    : "$MOOGSOFT_HOME/log/data-
capture/solarwinds_lam.log"
}
```

MySQL

Log location: `/var/log/mysqld.log`

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

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

```
log_warnings = 0
```

3. Restart the MySQL 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

Moogfarmd, LAMs and integrations 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 properties under `RollingFile` and `Policies` in `$MOOGSOFT_HOME/config/logging/<component_log_file_name>.log.json`

size

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

Type: Integer

Default: 500M

max

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/logging/<component_log_file_name>.log.json` as follows:

```
"Policies":
{
    "SizeBasedTriggeringPolicy":
    {
        "size": "500M"
    }
},
"DefaultRolloverStrategy":
{
    "max": "40"
}
```

Log Configuration File Examples

You can customize each configuration log file to control the behaviour of the logging for the different components in Cisco Crosswork Situation Manager.

See [Configure Logging](#) for more information on logging.

Default Configuration Files

The default log configuration file for servlets and utilities is as follows:

```
{
    "configuration": {
        "packages": "com.moogsoft",
        "monitorInterval": 2,
```



```
    "ThresholdFilter": {
      "level": "info"
    },
    "appenders": {
      "Console": {
        "name": "STDOUT",
        "PatternLayout": {
          "pattern": "%-5level: [%thread][%date{yyyMMdd
HH:mm:ss.SSS Z}] [%file:%line] +|%message|+%n"
        }
      }
    },
    "loggers": {
      "Logger": {
        "name": "com.moogsoft",
        "additivity": false,
        "AppenderRef": [{
          "ref": "STDOUT"
        }],
        "level": "info"
      }
    }
  }
}
```

The default log configuration file for Moogfarmd is:

```
{
  "configuration": {
    "packages": "com.moogsoft",
    "monitorInterval": 2,
    "ThresholdFilter": {
      "level": "info"
    },
    "appenders": {
      "Console": {
        "name": "STDOUT",
        "PatternLayout": {
          "pattern": "%-5level: [%thread][%date{yyyMMdd
HH:mm:ss.SSS Z}] [%file:%line] +|%message|+%n"
        }
      },
      "RollingFile": {
        "name": "FILE",
        "fileName": "${sys:MoogsoftLogFilename}",
        "filePattern": "${sys:MoogsoftLogFilename}-%d{MM-dd-yy}-
%i.gz",
        "PatternLayout": {
          "header": "${sys:MoogsoftLogHeader}",
          "pattern": "%-5level: [%thread][%date{yyyMMdd
HH:mm:ss.SSS Z}] [%file:%line] +|%message|+%n"
        }
      }
    }
  }
}
```



```
        "Policies": {
            "SizeBasedTriggeringPolicy": {
                "size": "500M"
            }
        },
        "DefaultRolloverStrategy": {
            "max": "40"
        }
    },
    "loggers": {
        "Logger": {
            "name": "com.moogsoft",
            "additivity": false,
            "AppenderRef": [{
                "ref": "${sys:MoogsoftLogAppender}"
            }],
            "level": "info"
        }
    }
}
```

Asynchronous Appender

You can configure a log file to use an asynchronous appender. This allows you to log event asynchronously. See [AsyncAppender](#) for details.

```
{
    "configuration": {
        "packages": "com.moogsoft",
        "monitorInterval": 2,
        "ThresholdFilter": {
            "level": "info"
        },
        "appenders": {
            "Console": {
                "name": "STDOUT",
                "PatternLayout": {
                    "pattern": "%-5level: [%thread][%date{yyyMMdd
HH:mm:ss.SSS Z}] [%file:%line] +|%message|+%n"
                }
            },
            "RollingFile": {
                "name": "FILE",
                "fileName": "${sys:MoogsoftLogFilename}",
                "filePattern": "${sys:MoogsoftLogFilename}-%d{MM-dd-yy}-
%i.gz",
                "PatternLayout": {
                    "header": "${sys:MoogsoftLogHeader}",
                    "pattern": "%-5level: [%thread][%date{yyyMMdd
HH:mm:ss.SSS Z}] [%c]: %message%n"
```



```
    },
    "Policies": {
      "SizeBasedTriggeringPolicy": {
        "size": "500M"
      }
    },
    "DefaultRolloverStrategy": {
      "max": "40"
    }
  },
  "Async" : {
    "name": "Async",
    "AppenderRef": {"ref": "FILE"}
  }
},
"loggers": {
  "Logger": {
    "name": "com.moogsoft",
    "additivity": false,
    "AppenderRef": [{
      "ref": "Async"
    }],
    "level": "info"
  }
}
}
```

Post Logs to Elasticsearch

You can configure logs to be posted to Elasticsearch using the "Http" section and the 'url' property direct them to an Elasticsearch server:

```
{
  "configuration": {
    "packages": "com.moogsoft",
    "monitorInterval": 2,
    "ThresholdFilter": {
      "level": "info"
    },
    "appenders": {
      "Console": {
        "name": "STDOUT",
        "PatternLayout": {
          "pattern": "%-5level: [%thread][%date{yyyMMddHH:mm:ss.SSS Z}] [%file:%line] +|%message|+%n"
        }
      },
      "RollingFile": {
        "name": "FILE",
        "fileName": "${sys:MoogsoftLogFilename}",
        "filePattern": "${sys:MoogsoftLogFilename}-%d{MM-dd-yy}-"
```



```
%i.gz",
    "PatternLayout": {
        "header": "${sys:MoogsoftLogHeader}",
        "pattern": "%-5level: [%thread][%date{yyyMMdd
HH:mm:ss.SSS Z}] [%file:%line] +|%message|+%n"
    },
    "Policies": {
        "SizeBasedTriggeringPolicy": {
            "size": "500M"
        }
    },
    "DefaultRolloverStrategy": {
        "max": "40"
    }
},
"http": {
    "name": "Elastic",
    "url": "http://localhost:9200/logs/farmdlogs/",
    "JsonLayout": {
        "compact": true,
        "eventEol": true,
        "locationInfo": true,
        "properties": true
    }
}
},
"loggers": {
    "Logger": {
        "name": "com.moogsoft",
        "additivity": false,
        "AppenderRef": [{
            "ref": "${sys:MoogsoftLogAppender}"
        }, {
            "ref": "Elastic"
        }],
        "level": "info"
    }
}
}
}
```

Save Logs to the Database

You can configure your logs to be saved to the database with a configuration similar to the following:

```
/*
To create the table for the logs to use:
CREATE TABLE IF NOT EXISTS logs(time TIMESTAMP, message TEXT, level TEXT,
logger TEXT, exception TEXT);
*/
```



```
{
  "configuration": {
    "packages": "com.moogsoft",
    "monitorInterval": 2,
    "ThresholdFilter": {
      "level": "info"
    },
    "appenders": {
      "Console": {
        "name": "STDOUT",
        "PatternLayout": {
          "pattern": "%-5level: [%thread][%date{yyyMMdd
HH:mm:ss.SSS Z}] [%file:%line] +|%message|+%n"
        }
      },
      "RollingFile": {
        "name": "FILE",
        "fileName": "${sys:MoogsoftLogFilename}",
        "filePattern": "${sys:MoogsoftLogFilename}-%d{MM-dd-yy}-
%i.gz",
        "PatternLayout": {
          "header": "${sys:MoogsoftLogHeader}",
          "pattern": "%-5level: [%thread][%date{yyyMMdd
HH:mm:ss.SSS Z}] [%file:%line] +|%message|+%n"
        },
        "Policies": {
          "SizeBasedTriggeringPolicy": {
            "size": "500M"
          }
        },
        "DefaultRolloverStrategy": {
          "max": "40"
        }
      },
      "JDBC" : {
        "name": "DB",
        "tableName": "logs",
        "DriverManager": {
          "connectionString":
"jdbc:mysql://localhost:3306/moogdb??useUnicode=true&characterEncoding=UTF-
8&characterSetResults=UTF-
8&connectTimeout=5000&rewriteBatchedStatements=true&cacheCallableStmts=true
&cachePrepStmts=true&callableStatementCacheSize=1000&prepStmtCacheSize=1000
&prepStmtCacheSqlLimit=100000&useCursorFetch=true&useSSL=false",
          "userName": "ermintrude",
          "password": "m00"
        },
        "Column": [{
          "name": "time",
          "isEventTimestamp": true
        }], {
```



```
        "name": "message",
        "pattern": "%message"
    }, {
        "name": "level",
        "pattern": "%level"
    }, {
        "name": "logger",
        "pattern": "%logger"
    }, {
        "name": "exception",
        "pattern": "%ex{full}"
    }
    ]
}
},
"loggers": {
    "Logger": {
        "name": "com.moogsoft",
        "additivity": false,
        "AppenderRef": [{
            "ref": "${sys:MoogsoftLogAppender}"
        }, {"ref": "DB"}],
        "level": "info"
    }
}
}
```

[Log Levels Reference](#)

Cisco Crosswork Situation Manager components [generate log files](#) to report their activity. Log messages sent from these components use the following log levels:

Level	Description
-------	-------------

ERROR	Errors have occurred but the application is still able to run.
WARN	Indicates something potentially serious or harmful has happened to your application.
INFO	Informational messages that report the application's normal behaviour.
DEBUG	Diagnostic and granular information that can be useful for debugging an application.
TRACE	Similar to DEBUG but even more fine-grained information.

[Configure Services to Restart](#)

You can use Service Manager to control process startup for Cisco Crosswork Situation Manager.



The utility can keep processes associated with Cisco Crosswork Situation Manager services alive if your system fails or restarts. For every new install of Cisco Crosswork Situation Manager, a cronjob entry deployed by the Moogsoft initialization script (`moog_init.sh`) runs the process manager script (`process_keepalive.sh`) every minute. This script attempts to restart the processes of any services listed in the process manager configuration file at `$MOOGSOFT_HOME/config/keep_procs_alive.conf`. By default, the following services are set to restart:

- RabbitMQ
- MySQL
- Nginx
- Elasticsearch
- Apache-Tomcat
- Moogfarmd.

You can either use the new `service_manager` utility to edit the configuration file, or edit the file manually (and enter a '1' for all services you want to restart and enter a '0' for those that you want left alone). If you have a custom LAM, you can add this to the configuration file with the flag to determine the restart behavior. For example:

```
Customlamd=1
```

If your system fails or restarts, the Service Manager utility automatically starts after one minute and attempts to restart the configured processes/services up to three times. If the service does not start after three attempts, the utility disables the service restart attempts in future.

Service Manager is useful for ensuring non-core processes start and run if Cisco Crosswork Situation Manager fails or restarts. For example, you might want to ensure specific LAMs remain alive so no events are missed if your system reboots.

Configure Service Manager

You can configure Service Manager to control which services and their associated processes to start up when Cisco Crosswork Situation Manager reboots:

1. Run the Service Manager:

```
$MOOGSOFT_HOME/bin/utlis/service_manager
```

The default utility settings are shown as follows:

```
----- Page [1/3] ---
#                               Service Manager                               #
-----
Check the services that will be started and kept alive...
```



```
[x] rabbitmq
[x] mysql
[x] nginx
[x] elasticsearch
[x] apache-tomcat
[x] moog_farmd
[ ] ansibletower_lam
[ ] appdynamics_lam
[ ] aws_lam
[ ] azure_classic_lam
[ ] azure_lam
[ ] ca_spectrum_lam
[ ] datadog_client_lam
[ ] datadog_lam
[ ] dynatrace_apm_lam
[ ] dynatrace_apm_plugin_lam
[ ] dynatrace_notification_lam
[ ] dynatrace_synthetic_lam
[ ] email_lam
[ ] emc_smarts_lam
[ ] extrahop_lam
[ ] fluentd_lam
[ ] hp_nnmi_lam
[ ] hp_omi_lam
```

Use arrows, page up/down and ENTER to navigate

2. Navigate through the list of available services using the directional arrows. There are multiple pages to scroll through.
3. Press **Space** or **Enter** to add or remove services you want to restart or do not want to restart. An [x] appears next to any services you select.
4. You can enable or disable Service Manager from restarting all services on the last page.
5. Select 'Apply Changes' and press **Enter** to make the changes.

After you exit, the `process_keepalive.sh` script keeps selected services alive if Cisco Crosswork Situation Manager fails or restarts.

Service Manager Command Line Reference

You can also configure the Service Manager using command line arguments. The utility uses the following syntax:

```
service_manager --service=<service_name> --command=<action>
```

The Service Manager resides at `$MOOGSOFT_HOME/bin/utils/service_manager`. You can configure the utility using the following arguments:

Argument	Input	Description
----------	-------	-------------



<code>-h, --help</code>	<code>-</code>	Displays the syntax and arguments available in Service Manager.
<code>-s, --service=</code>	<code>service name</code>	<p>Name of the service you want to execute the Service Manager command on.</p> <p>Apart from the core services, these include all of the LAMs. For example: <code>email_lam</code>, <code>rest_lam</code>, <code>trapd_lam</code> etc.</p>
<code>-c, --command=</code>	<code>enable disable enable_start disable_stop</code>	<p>Commands the Service Manager can execute against the specified service:</p> <ul style="list-style-type: none">• <code>enable</code> - Enables the service auto start option.• <code>disable</code> - Disables the service auto start option.• <code>enable_start</code> - Starts and enables the service.• <code>disable_stop</code> - Stops and disables the service.
<code>-a, --autostart-all=</code>	<code>enable disable</code>	Disables or enables the auto start option for all services.

You can only run a command against a single service at a time using the command line arguments. For example, if you wanted to enable the Service Manager to restart the Email LAM in the event of a failover:

```
service_manager --service=email_lam --command=enable
```

Disable Service Restart

There are two ways to disable the default service restart functionality.

Disable the `process_keepalive.sh` cronjob by removing it from the cron table, the commands scheduled to run on your system. To edit the cron table:

```
crontab -e
```

Delete or comment out `/usr/share/moogsoft/bin/utils/process_keepalive.sh 2>&1` from the file.

Alternatively, disable the Service Manager utility:

```
$MOOGSOFT_HOME/bin/utils/service_manager -a disable
```



Service Manager Logging

To check the Service Manager logs you can view:

```
/var/log/moogsoft/process/keepalive.log
```

This contains all logs relating to the Service Manager utility (`service_manager`) and to the process that attempts to keep the services alive (`process_keepalive`).

Configure the Tool Runner

The Tool Runner uses ssh to run tools and integrations. You must edit the servlets configuration file in Cisco Crosswork Situation Manager in order to use the Tool Runner in the UI.

Before You Begin

Before you begin to configure the Tool Runner, ensure you have met the following requirements:

- You have created or identified an operating system user that you will use to run tools.
- You have the permissions to modify Cisco Crosswork Situation Manager configuration files.
- You have set the `PasswordAuthentication` property to `yes` in the `/etc/ssh/sshd_config` file on the Cisco Crosswork Situation Manager server and restarted the `sshd` service.

Configure the Tool Runner

To manually configure the Tool Runner, edit the Servlets configuration file located at `$MOOGSOFT_HOME/config/servlets.conf` as follows:

1. Update these properties in the `toolrunner` section of the file:
 - **Toolrunnerhost:** The Tool Runner host to run on. If running a distributed installation this is the host name of the machine where Apache Tomcat is installed.
 - **Toolrunneruser:** The Tool Runner user name. The user must exist in your system and have the appropriate permissions to run the required tools.
 - **Toolrunnerpassword:** The Tool Runner user password.
 - **Encrypted_toolrunnerpassword:** An encrypted Tool Runner password. Use either the password or encrypted password property.
 - `Execute_locally:`
 - `Webhost:`



- **Sshtimeout**: SSH timeout period in seconds.

2. Restart Apache Tomcat.

3. Restart Moogfarmd.

Once you have completed the configuration, Tool Runner is available in the Cisco Crosswork Situation Manager UI.

An example toolrunner section in the servlets configuration file is as follows:

```
toolrunner:
{
    toolrunnerhost                : "localhost",
    toolrunneruser                 : "moogtoolrunner",
    toolrunnerpassword             : "moogtoolrunner",
    #encrypted_toolrunnerpassword :
    "rmW2daCwMyI8JGZygfEJj0MZdbIkUqX3tT/OIVfMGyI=",
    #execute_locally               : false,
    #webhost                       :
    "https://localhost",
    sshtimeout                     : 900000
}
```

Script to Create a Tool Runner User

To create a script that automatically creates a tool runner user with the default configuration, copy the following code into a file and save it. For example, to a script file named 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
```

Run the script on a clean installation 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, for example, 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**).

Using Server Tools in the Cisco 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');
```

Tool Arguments

The arguments 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/ssh/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.



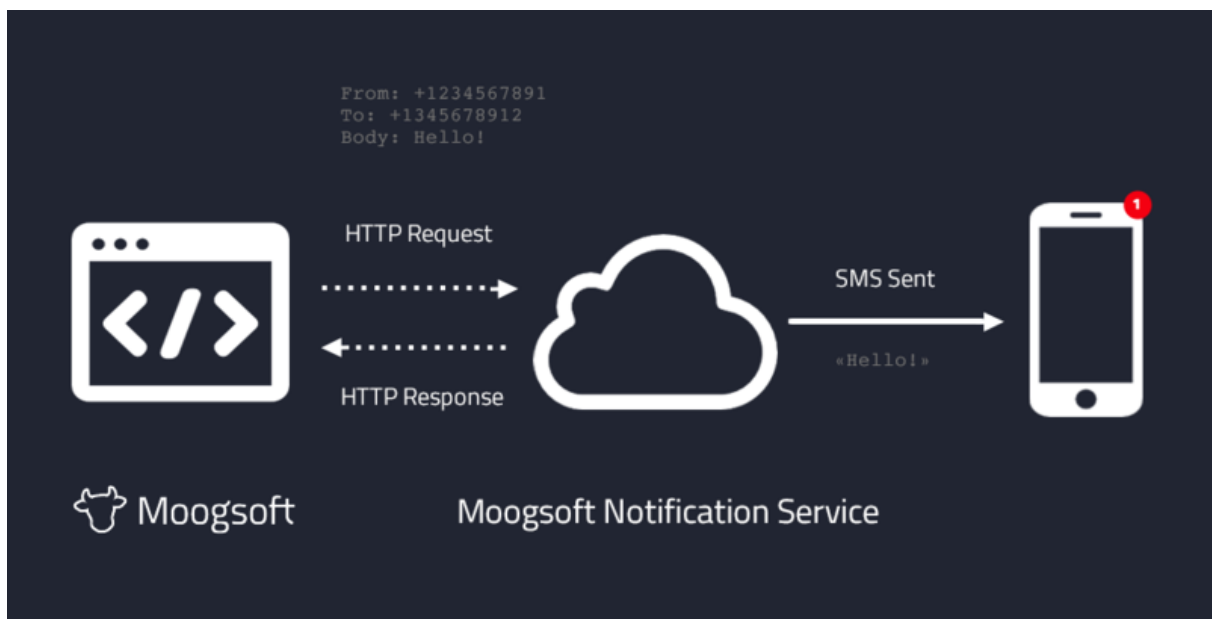
3. Confirm that you can now login to the remote SSH server without a password: `ssh user@remotehost.com`.
4. Copy the private key (`id_rsa`) to a directory called **keys**, which you need to create in the tomcat home directory, for example, `/export/apache-tomcat-7.0.29`.

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 suders file => `user ALL=(ALL) NOPASSWD: ALL`

Configure SMS Notifications

Cisco Crosswork Situation Manager for Mobile uses a cloud-based notification service that allows SMS messages to be sent by its REST API.



Any requests to the notification service's REST API require authentication. For HTTP basic authentication, the username is set to your AccountSid and the password is set to your AuthToken. See step 2 in the configuration instructions below.

Sent text messages are limited to 1600 characters in length. Any messages longer than this will be truncated.



Configuration

Follow these steps to configure the mobile version of Cisco Crosswork Situation Manager to send and receive SMS messages:

1. In the Cisco Crosswork Situation Manager UI, navigate to **System Settings > Users**. Add phone numbers for all users that will receive SMS messages.

Note

All phone numbers must include the international country code in [E.164](#) format. For example +1, +44, +61.

2. Contact [Moogsoft Support](#) or your Cisco Sales Engineer to enable SMS. You will be provided with an account session ID, authentication token and a phone number.
3. Open a terminal session to your Cisco Crosswork Situation Manager server and edit the file `$MOOGSOFTHOME/config/system.conf`. Add your details to this file in the following format:

```
"sms_sender":
{
    "account_sid"           :
    "AB1234c6d91dfe4eef9b3899dkrw34aabbab2b",
    "auth_token"            : "4eee23d50de54333f6949366fe0882a4",
    "phone_number"          : "+12345678910",
    "character_limit"       : 1600
}
```

4. Restart Apache Tomcat to activate the changes. See [Control Moogsoft AIOps Processes](#) for details.

By default you will receive SMS notifications about all invitations and assignments. This cannot be changed at present.

Enable Situation Room Plugins

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.

Note

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	MUL	NULL	

```
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 `$value` dynamic 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.

Import a 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:

1. Import the .csv topology file into the databases using the topology builder found at `$MOOGSOFT_HOME/bin`:

```
./topology_builder -t <file_name>.csv
```

The `-t` option defines the topology file. You can use `-l` to define the level of debug output. For all options see [Topology Builder Command Reference](#).

2. 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_catalog` table. By default topology builder assigns each node to the 'Network' group and the 'Unix servers' competency. The utility records the topological information in the `moog_reference.one_hop_topo` and `moog_reference.topo_nodes` tables.

After you have imported your topological data, you can use the values in clustering calculations by 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 `-r <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 the [Topology Builder](#). The Topology Builder command-line utility accepts the following arguments:

Argument	Input	Description
<code>-h, --help</code>	-	Display the topology_builder utility syntax and option descriptions.
<code>-l, --loglevel <arg></code>	WARN INFO DEBUG TRACE	Log level controlling the amount of information that topology_builder logs. Defaults to INFO.
<code>-r, --rebuild</code>	Integer <percentage>	Percentage of unprocessed topology before a rebuild. Utility rebuilds



<arg>

topology when the percentage of unlabelled nodes exceeds this value. If no percentage value is entered, the topology is not rebuilt.

-t, --topology-
file
<csv_filename>

Name of the .csv file containing the pairs of connected nodes with an optional weighting value.

Configure Historic Data Retention

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.

Note

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 [Historic Database Benefits](#) 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: Graze API MoogDb V2

- `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 Moogfarmd process.

When the splitting process is active, you can monitor its progress in the 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 the 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:	→ Split occurs	Situation 3 (open) with member alerts:	Situation 1 (closed) with member alerts:
• Alert 1 (closed)		• Alert 5 (closed)	• Alert 1 (closed)
• Alert 2 (closed)		• Alert 6 (closed)	• Alert 2 (closed)
• Alert 3 (closed)		• Alert 7 (closed)	• Alert 3 (closed)
Situation 2 (closed) with member alerts:		• Alert 8 (open)	Situation 2 (Closed) with member alerts:
• Alert 4 (closed)		Situation 4 (Open) with member alerts:	• Alert 4 (closed)
• Alert 5 (closed)		• Alert 8 (open)	• Alert 5 (closed)
• Alert 6 (closed)		• Alert 9 (open)	• Alert 6 (closed)
Situation 3 (open) with member alerts:		Loose alerts:	Loose alerts:
• Alert 5 (closed)		• Alert 11 (open)	• Alert 10 (closed)
• Alert 6 (closed)			Other alerts:
• Alert 7 (closed)			• Alert 7 (closed)
• Alert 8 (open)			



Situation 4 (open) with member alerts:

- Alert 8 (open)
- Alert 9 (open)

Loose alerts:

- Alert 10 (closed)
- Alert 11 (open)

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 [Configure Historic Data Retention](#) for further details.

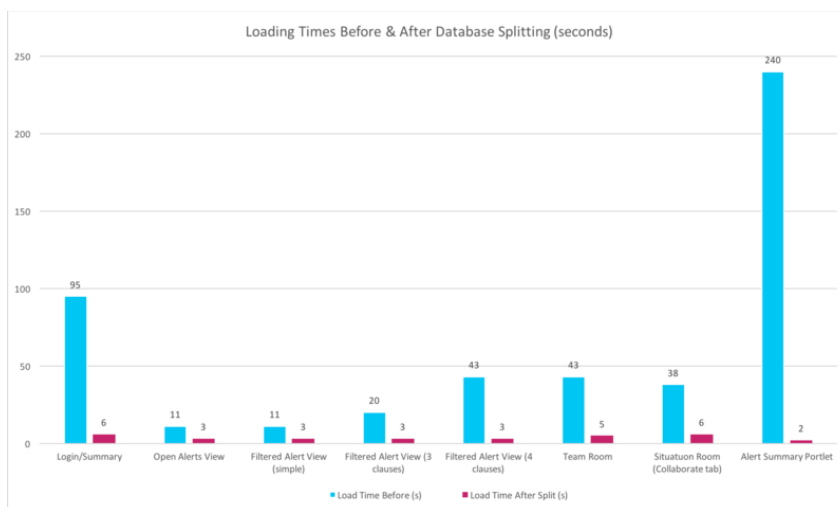
Performance Improvement Metrics

Cisco engineering used the following baselines for testing performance differences between split-database and single-database systems:

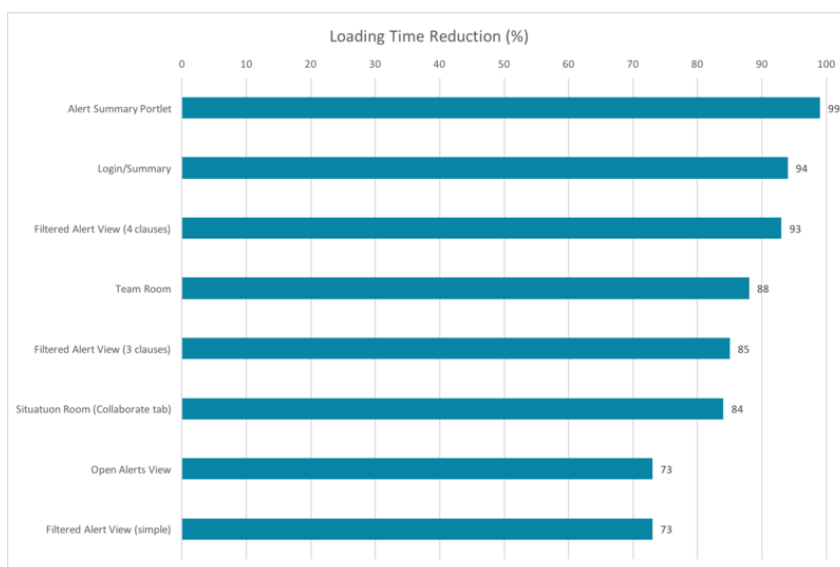


- Single Linux server with 64 x 2.3GHz cores
- 128 GB RAM
- 13 months of data from a very large system (100M events, 14M alerts and 1M Situations)
- 3% open alerts and Situations.

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 [Configure Historic Data Retention](#) feature and evaluate its benefits:

- Read the documentation: [Configure Historic Data Retention](#)
- 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 [Configure Historic Data Retention](#). 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.

<code>-e, --enable <arg></code>	String	Enable the retention of historic data in a separate database.
<code>-g, --grace_period <arg></code>	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).
<code>-h, --help</code>	-	Display the syntax and option descriptions.
<code>-l, --loglevel <arg></code>	WARN INFO DEBUG TRACE	Specify the output verbosity. Defaults to INFO.
<code>-n, --database_name <arg></code>	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.
<code>-p, --password <arg></code>	String	Password for the MySQL user specified in the user argument -u. Defaults to ''.
<code>-r, --run_interval <arg></code>	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).
<code>-s, --sigs_batch_size <arg></code>	Integer: <number of Situations>	Number of Situations to include in each batch. Defaults to 500.
<code>-u, --user <arg></code>	String	MySQL username with schema creation privileges. Defaults to root.

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/archived` directory.

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
<code>-a, --alert_filter <arg></code>	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.
<code>-b, --update_batch_size <arg></code>	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.
<code>-d,--delimiter <arg></code>	String	Defaults to comma ",". Delimiter to insert between values in the export file.
<code>-e,--export</code>	-	Export the data to a file.
<code>-f,--file_age <arg></code>	Integer: <number of days>	Delete files from the default directory /usr/local/archived that are older than the specified number of days.
<code>-g,--loglevel <arg></code>	WARN INFO DEBUG TRACE	Specify the output verbosity. Defaults to INFO.
<code>-h,--help</code>	-	Display the moog-archiver utility syntax and option descriptions.
<code>-i,-- situation_filter <arg></code>	-	Include all Situations that match the specified global alert filter.
<code>-l,-- loose_alert_age <arg></code>	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 Configure Historic Data Retention is enabled.
<code>-m,-- include_statistics</code>	-	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.
<code>-n,-- statistics_age <arg></code>	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 Configure Historic Data Retention is enabled.
<code>-o,--retain_open</code>	-	Only include data from Situations with Closed, Dormant or Superseded status. Cannot be used with the Situation filter -i.



<code>-p,--archive_path</code> <code><arg></code>	String: <code><path></code>	Defaults to <code>/usr/local/archived</code> . Destination path for the archived data.
<code>-r,--remove</code>	-	Delete data from the database.
<code>-s,--situation_age</code> <code><arg></code>	Integer: <code><number of days></code>	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 Configure Historic Data Retention is enabled.
<code>-t,--loose_alerts_only</code>	-	Include loose alerts only. Cannot be used with <code>-i -o -s</code>
<code>-y,--delay_time</code> <code><arg></code>	Integer: <code><number of milliseconds></code>	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.
<code>-z,--id_batch_size</code> <code><arg></code>	Integer: <code><number of rows></code>	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.

Probable Root Cause

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 resolves Situations quickly by examining the 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 [Configure and Retrain Probable Root Cause](#) for more information on training your model. [Configure and Retrain Probable Root Cause](#)

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.

URL-based Filters

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	<code>https://<localhost>/</code>	Host name of your Cisco Crosswork Situation Manager instance.
The view type	<code>[/alerts /situations /situationalerts]</code>	Defines whether an alert view, Situation view or alerts assigned to Situations are displayed.
The filter type	<code>?[filtereditor=basic filtereditor=advanced]</code>	Defines whether the filter will use basic or advanced query syntax.
<hr/>		
Note		
Please note: Place a question mark (?) at the start of your query parameter and separate each subsequent parameter with an ampersand (&)		
The parameters	<code>&[filter-active_sig_list= filter-alert_id= filter-agent=]</code>	These are the parameters, operators and values used to define in the filter. E.g. <code>&filter-alert_id=12, &filter-count=5</code> etc. For all available parameters click here .
<hr/>		
Note		
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://<localhost>/#/situationalerts/172?filtereditor=basic&filter-type=CPUHigh&filter-severity=2
```

A breakdown of this URL's components are described in the table below:

Example Component	Description
<code>https://<localhost>/</code>	Host name of your Cisco Crosswork Situation Manager instance.
<code>/#/situationalerts/172</code>	Specifies an alert view of all alerts assigned to Situation 172.
<code>?filtereditor=basic</code>	Specifies that you want to use basic filter query string.
<code>&filter-type=CPUHigh&filter-severity=2</code>	Defines the filter will display all alerts with the alert type 'CPUHigh' and that have the severity of 'Warning' (severity level 2).



Type: CPUHigh Severity: Warning Filter						
	SEVERITY ↓	TYPE	FIRST EVENT TIME	LAST EVENT TIME	COUNT	DESCRIPTION
<input type="checkbox"/>	Warning	CPUHigh	22:01:09 20/04/20...	09:52:59 24/04/20...	99	CPU Exceeds 90%
<input type="checkbox"/>	Warning	CPUHigh	22:01:09 20/04/20...	09:53:00 24/04/20...	104	CPU Exceeds 90%
<input type="checkbox"/>	Warning	CPUHigh	22:01:09 20/04/20...	09:53:00 24/04/20...	106	CPU Exceeds 90%
<input type="checkbox"/>	Warning	CPUHigh	22:01:08 20/04/20...	09:52:52 24/04/20...	100	CPU Exceeds 90%
<input type="checkbox"/>	Warning	CPUHigh	22:01:08 20/04/20...	09:52:56 24/04/20...	105	CPU Exceeds 90%

Note

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.

Note

Please note: For more information on adding custom_info fields see [Custom Info](#).

The example below shows a URL-based filter

```
https://<localhost>/#/situations?filtereditor=basic&filter-  
custom_info.something_new=5
```

Component	Description
https://<localhost>/	Host name of your Cisco Crosswork Situation Manager instance.
/#/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](#).

Note

***Please note:** To encode a filter query, open DevTools (right-click and select **Inspect**) and go to **Console**:

Type 'encodeURIComponent', insert the query in brackets and then double quotation marks:

Console Entry

```
encodeURIComponent ("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://<localhost>/#/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://<localhost>/	Host name of your Cisco Crosswork Situation Manager instance.
#/situationalerts/172	Directs the filter to a view of all alerts assigned to Situation 172.
?filtereditor=advanced	Specifies that you want to use advanced filter query syntax.



&filter-
query=Severity%20=%20'Critical'%20AND%20Severity%20=%20'Minor'

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.coi	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.lspat.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/??filtereditor=advanced&filter-query=%60Something%20new%60%20MATCHES%20%225%22>

Example Component	Description
<a href="https://<localhost>/#/situations/??filtereditor=advanced&filter-query=%60Something%20new%60%20MATCHES%20%225%22">https://<localhost>/	Host name of your Cisco Crosswork Situation Manager instance.
#/situations?	Directs the filter to a Situations view.
?filtereditor=advanced	Specifies that you want to use advanced filter query string.
&filter-query=%60Something%20new%60%20MATCHES%20%225%22	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.

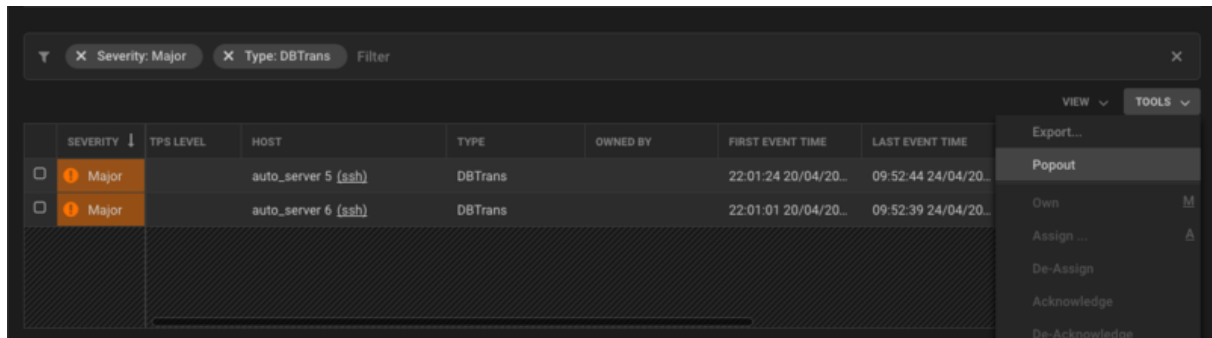


Note

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**:



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&filtereditor=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
```

Note

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?filtereditor=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?filtereditor=basic&filter-service_list=$service_list
```

Parameters

The tables below list the available parameters and the associated operators for alerts and Situations:

Warning

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 Paramaters

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

System Configuration

You can configure the various components of Cisco Crosswork Situation Manager using the system configuration file. These include:

- Message Bus
- Databases
- Search
- Failover
- Process monitoring



- Web host address
- Logging

Configure your System

Edit the configuration file to control the behavior of the different components in your Cisco Crosswork Situation Manager system. You can find the file at `$MOOGSOFT_HOME/config/system.conf`.

See the [System Configuration Reference](#) for a full description of all properties. Some properties in the file are commented out by default. Uncomment properties to configure and enable them.

Message Bus

You can edit your Message Bus and RabbitMQ configuration in the `themooms` section. It allows you to:

- Configure your Message Bus zones and brokers.
- Control and minimize message loss during a failure.
- Control how senders handle Message Bus failures.
- Control what happens during periods of extended Message Bus unavailability.
- Configure the SSL protocol you want to use.

For more information see the [Message Bus](#) documentation.

Database

You can edit your database configuration in the `mysql` section:

1. Configure your host name, database names and database credentials:
 - **host**: Name of your host.
 - **moogdb_database_name**: Name of the Moogdb database.
 - **referencedb_database_name**: Name of the Cisco Crosswork Situation Manager reference database.
 - **username**: Username for the MySQL user that accesses the database.
 - **encrypted_password**: Encrypted password for the MySQL user.
 - **password**: Password for the MySQL user.
 - **port**: Default port that Cisco Crosswork Situation Manager uses to connect to MySQL.
2. Configure the port, deadlock retry attempts and multi-host connections:



- **maxRetries:** Maximum number of retries in the event of a MySQL deadlock.
 - **retryWait:** Number of milliseconds to wait between each retry attempt.
 - **failover_connections:** Hosts and ports for the different servers that are connected to the main host.
3. Configure the SSL connections to the MySQL database:
- **trustStorePath:** Path to location that stores the server certificate.
 - **trustStoreEncryptedPassword:** Path to location that stores your encrypted trustStore password.
 - **trustStorePassword:** Path to location that stores your trustStore password.

Elasticsearch

You can edit your search configuration in the search section:

1. Configure the Elasticsearch connection timeouts:
 - **connection_timeout:** Length of time in milliseconds before the connection times out.
 - **request_timeout:** Length of time in milliseconds before the request times out.
2. Configure the Elasticsearch limit and nodes:
 - **limit:** Maximum number of search results that Elasticsearch returns from a search query.
 - **nodes:** Hosts and ports for the different Elasticsearch servers connected in a cluster.

Failover

You can edit failover configuration in the failover section:

1. Configure persistence in the event of a failover:
 - **persist_state:** Enable or disable the persistence of the state of all Moolets in the event of a failover.
2. Configure the [Hazelcast](#) cluster, this is Cisco Crosswork Situation Manager implementation of persistence:
 - **network_port:** Port to connect to on each specified host.
 - **auto_increment:** Enable for Hazelcast to attempt to the next incremental available port number if the configured port is unavailable.
 - **hosts:** List of hosts that can participate in the cluster.



- **man_center:** Configures the cluster information that you can view in the Hazelcast Management Center UI.
 - **cluster_per_group:** Enable the stateful information from each process group to persist in a dedicated Hazelcast cluster.
3. Configure failover options that apply to Moogfarmd and the LAMs:
- **keepalive_interval:** Time interval in seconds at which processes report their active/passive status and check statuses of other processes.
 - **margin:** Amount of time in seconds after `keepalive_interval` before Cisco Crosswork Situation Manager considers processes that do not report their status to be dead.
 - **failover_timeout:** Number of seconds to wait for previously active process to become passive during a manual failover.
 - **automatic_failover:** Allow a passive process to automatically become active if no other active processes are detected in the same process group.
 - **heartbeat_failover_after:** Number of consecutive heartbeats that a process fails to send before Moogfarmd considers it inactive.

Process Monitor

You can edit the process monitor configuration in the `process_monitor` section:

1. Configure the heartbeat interval and delay:
 - **heartbeat:** Interval in milliseconds between heartbeats sent by processes.
 - **max_heartbeat_delay:** Number of milliseconds to wait before declaring heartbeat as missing.
2. Configure the Moogfarmd and which processes you can control from the UI:
 - **group:** Name of the group of processes and subcomponent processes that you want to control from the UI.
 - **instance:** Name of the instance of Cisco Crosswork Situation Manager you want to configure.
 - **service_name:** Name of the service you want to control.
 - **process_type:** Type of process you want to control.
 - **reserved:** Determines if Cisco Crosswork Situation Manager considers the process as critical in process monitoring.

Encryption

You can edit the encryption configuration in the `encryption` section.



Configure the location of the encryption key file:

- **encryption_key_file**: Default location of the encryption key file.

High Availability

You can edit the high availability configuration in the ha section.

Configure the default HA cluster name:

- **cluster**: Default HA cluster name.

Web Hostname

You can edit the hostname configuration in the webhost section.

Configure the hostname:

- **webhost**: Default web host address.

Service Port Range

You can edit the port range that Cisco Crosswork Situation Manager services use when they look for open ports.

Configure the port range used by services:

- **port_range_min**: Minimum port number in the range.
- **port_range_max**: Maximum port number in the range.

Example

The following example shows `system.conf` with the default configuration and all available properties enabled:

```
{
    "mooms": {
        "zone": "",
        "brokers": [{
            "host": "localhost",
            "port": 5672
        }],
        "username": "moogsoft",
        "password": "m00gs0ft",
        "encrypted_password":
        "e5u00LY3HQJZCltG/caUnVbxVN4hImm4gIOpb4rwpF4=",
        "threads": 10,
        "message_persistence": false,
        "message_prefetch": 100,
        "max_retries": 100,
        "retry_interval": 200,
        "cache_on_failure": false,
        "cache_ttl": 900,
    }
}
```



```
        "confirmation_timeout": 2000,
        "ssl": {
            "ssl_protocol": "TLSv1.2",
            "server_cert_file": "server.pem",
            "client_cert_file": "client.pem",
            "client_key_file": "client.key"
        }
    },
    "mysql": {
        "host": "localhost",
        "moogdb_database_name": "moogdb",
        "referencedb_database_name": "moog_reference",
        "username": "ermintrude",
        "encrypted_password":
"vQj7/yom7e5ensSEb10v2Rb/pgkaPK/40cUlEjYNtQU=",
        "password": "m00",
        "port": 3306,
        "maxRetries": 10,
        "retryWait": 50,
        "failover_connections": [{
            "host": "193.221.20.24",
            "port": 3306
        },
        {
            "host": "143.47.254.88",
            "port": 3306
        },
        {
            "host": "234.118.117.132",
            "port": 3306
        }
    ],
        "ssl": {
            "trustStorePath": "etc/truststore",
            "trustStoreEncryptedPassword":
"vQj7/yom7e5ensSEb10v2Rb/pgkaPK/40cUlEjYNtQU=",
            "trustStorePassword": "moogsoft"
        }
    },
    "search": {
        "connection_timeout": 1000,
        "request_timeout": 10000,
        "limit": 1000,
        "nodes": [{
            "host": "localhost",
            "port": 9200
        }
    ]
    },
    "failover": {
```



```
"persist_state": false,
"hazelcast": {
  "network_port": 5701,
  "auto_increment": true,
  "hosts": ["localhost"],
  "man_center": {
    "enabled": false,
    "host": "localhost",
    "port": 8091
  },
  "cluster_per_group": false
},
"keepalive_interval": 5,
"margin": 10,
"failover_timeout": 10,
"automatic_failover": false,
"heartbeat_failover_after": 2
},
"process_monitor": {
  "heartbeat": 10000,
  "max_heartbeat_delay": 1000,
  "processes": [{
    "group": "moog_farmd",
    "instance": "",
    "service_name": "moogfarmd",
    "process_type": "moog_farmd",
    "reserved": true,
    "subcomponents": [
      "AlertBuilder",
      "Default Cookbook",
      "TeamsMgr",
      "Housekeeper",
      "AlertRulesEngine",
      "SituationMgr",
      "Notifier"
    ]
  },
  {
    "group": "servlets",
    "instance": "",
    "service_name": "apache-tomcat",
    "process_type": "servlets",
    "reserved": true,
    "subcomponents": [
      "moogsvr",
      "moogpoller",
      "toolrunner",
      "situation_similarity"
    ]
  }
],
{
```



```
        "group": "logfile_lam",
        "instance": "",
        "service_name": "logfilelamd",
        "process_type": "LAM",
        "reserved": false
    },
    {
        "group": "rest_lam",
        "instance": "",
        "service_name": "restlamd",
        "process_type": "LAM",
        "reserved": false
    },
    {
        "group": "socket_lam",
        "instance": "",
        "service_name": "socketlamd",
        "process_type": "LAM",
        "reserved": false
    },
    {
        "group": "trapd_lam",
        "instance": "",
        "service_name": "trapdlamd",
        "process_type": "LAM",
        "reserved": false
    },
    {
        "group": "rest_client_lam",
        "instance": "",
        "service_name": "restclientlamd",
        "process_type": "LAM",
        "reserved": false
    }
]
},
"encryption": {
    "encryption_key_file": "/location/of/.key"
},
"ha": {
    "cluster": "MOO"
},
"webhost": "localhost",
"port_range_min": 50000,
"port_range_max": 51000
}
```

Start and Stop Moogfarmd



Restart the Moogfarmd service to activate any changes you make to the system configuration file.

The service name is moogfarmd.

See [Control Moogsoft AIOps Processes](#) for further details.

System Configuration Reference

This is a reference for the [system configuration](#) file which is located at `$MOOGSOFT_HOME/config/system.conf`

It contains the following sections and properties:

Message Bus (MooMs)

Configuration properties relating to the Message Bus, also known as MooMs:

zone

Name of the zone.

Type: String

Required: No

Default: Empty

brokers

Hostname and port number of the RabbitMQ broker:

Type: Array

Required: No

Default: [{ "host" : "localhost", "port" : 5672 }]

username

Username of the RabbitMQ user. This needs to match the RabbitMQ broker configuration. If commented out, it uses the default "guest" user.

Type: String

Required: No

Default: "guest"

password

Password for the RabbitMQ user. You can choose to either have a password or an encrypted password, you cannot use both.

Type: String



Required: Yes. If you are not using encrypted password.

Default: "guest"

encrypted_password

Encrypted password for the RabbitMQ user. You can choose to either have a password or an encrypted password, you cannot use both. See [Moog Encryptor](#) if you want to encrypt your password.

Type: String

Required: Yes. If you are not using password.

Default: N/A

threads

Number of threads a process can create in order to consume the messages from the Message Bus. If not specified, the thread limit = (Number of processors x 2) + 1. Altering this limit affects the performance of Cisco Crosswork Situation Manager processes such as Moogfarmd and Moogpoller.

If your logs indicate an issue in creating threads, Cisco advises that you increase the ulimit, the maximum number of file descriptors each process can use, for the Cisco Crosswork Situation Manager user. You can set this limit in `/etc/security/limits.conf`.

Type: Integer

Required: No

Default: "10"

message_persistence

Controls whether RabbitMQ persists importance messages or not. Message queues are durable by default and data is replicated between nodes in High Availability mode. Setting this value to false, means that replicated data is not stored to disk.

Type: Boolean

Required: No

Default: "true"

message_prefetch

Controls how many messages a process can take from the Message Bus and store in memory as a buffer for processing. This configuration allows processes to regulate message consumption which can ease backlog and memory consumption issues. The higher the number, the more messages held in the process's memory.

Type: Integer



Required: No

Default: "0"

max_retries

Maximum number of attempts to resend a message that failed to send. Cisco Crosswork Situation Manager only attempts a retry when there is a network outage or if `cache_on_failure` is enabled.

You can use this in conjunction with the `retry_interval` property. For example, a combination of 100 maximum retries and 200 milliseconds for retry interval leads to a total of 20 seconds. The combined default value for these properties was chosen to handle the typical time for a broker failover in a clustered environment.

Type: Integer

Required: No

Default: "100"

retry_interval

Maximum length of time to wait in milliseconds between each attempt to retry and send a message that failed to send.

You can use this in conjunction with the `max_retries` property. The combined value for these properties was chosen to handle the typical time for broker failover in a clustered environment.

Type: Integer

Required: No

Default: "100"

cache_on_failure

Controls whether Cisco Crosswork Situation Manager caches the message internally and resends it if there is an initial retry failure. The system attempts to resend any cached messages in the order they were cached until the time-to-live value, defined by the `cache_ttl` property, is reached.

Type: Boolean

Required: No

Default: "false"

cache_ttl

Length of time in seconds that Cisco Crosswork Situation Manager keeps cached messages in the cache list before discarding them. If a message is not successfully resent within this timeframe it is still discarded.



This defaults to 900 seconds (15 minutes). Increasing this value has a direct impact on sender process memory.

Type: Integer

Required: No

Default: "900"

confirmation_timeout

Length of time in milliseconds to wait for the Message Bus to confirm that a broker has received a message. Cisco does not advise changing this value.

Type: Integer

Required: No

Default: "2000"

Message Bus SSL

ssl_protocol

SSL protocol you want to use. JRE 8 supports "TLSv1.2", "TLSv1.1", "TLSv1" or "SSLv3".

Type: String

Required: No

Default: "TLSv1.2"

server_cert_file

Path to the directory that contains the SSL certificates. You can use a relative path based upon the \$MOOGSOFT_HOME directory. For example, config indicates \$MOOGSOFT_HOME/config.

Type: String

Required: No

Default: "server.pem"

client_cert_file

Enables client authentication if you provide a client certificate and key file.

Type: String

Required: No

Default: "client.pem"

client_key_file



Enables client authentication if you provide a client key file. The file must be in [PKCS#8](#) format.

Type: String

Required: No

Default: "client.key"

MySQL

host

Host name or server name of the server that is running MySQL.

Type: String

Required: No

Default: "localhost"

moogdb_database_name

Name of the primary Cisco Crosswork Situation Manager database.

Type: String

Required: No

Default: "moogdb"

reference_database_name

Name of the Cisco Crosswork Situation Manager reference database.

Type: String

Required: No

Default: "moog_reference"

username

Username of the MySQL user.

Type: String

Required: No

Default: "ermintrude"

encrypted_password

Encrypted password for the MySQL user. See [Moog Encryptor](#) if you want to encrypt your password.

Type: String



Required: No

Default: N/A

password

Password for the MySQL user.

Type: String

Required: No

Default: "m00"

port

Port that MySQL uses.

Type: Integer

Required: No

Default: "3306"

maxRetries

Maximum number of MySQL query retries to attempt in the event of a deadlock.

Type: Integer

Required: No

Default: "10"

retryWait

Length of time in milliseconds to wait between retry attempts.

Type: Integer

Required: No

Default: "50"

failover_connections

Hosts and ports for the different servers that are connected to the main host. For example, master-master, master-slave. In the event of connection failover, the connection cannot be read-only (slave).

Type: List

Required: No

Default: N/A

MySQL SSL



trustStorePath

Path to the directory that contains the trustStore you want to use for SSL connections to your MySQL database. You can use a relative path based upon the \$MOOGSOFT_HOME directory. For example, config indicates \$MOOGSOFT_HOME/config/truststore.

Type: String

Required: No

Default: "etc/truststore"

trustStoreEncryptedPassword

Your encrypted trustStore password. See [Moog Encryptor](#) if you want to encrypt your password.

Type: String

Required: No

Default: "vQj7/yom7e5ensSEb10v2Rb/pgkaPK/40cU1EjYNtQU="

trustStorePassword

Your trustStore password.

Type: String

Required: No

Default: "moogsoft"

Search

connection_timeout

Length of time in milliseconds before the connection to the Elasticsearch server times out.

Type: Integer

Required: No

Default: "1000"

request_timeout

Length of time in milliseconds before an Elasticsearch request times out.

Type: Integer

Required: No

Default: "1000"



limit

Maximum number of search results that Elasticsearch returns.

Type: Integer

Required: No

Default: "1000"

nodes

Hosts and ports for the different Elasticsearch servers connected in a cluster.

Type: Array

Required: No

Default: [{"host" : "localhost", "port" : 9200}]

Failover

persist_state

Enable or disable the persistence of the state of all Moolets in the event of a failover.

Type: Boolean

Required: No

Default: "false"

network_port

Port to connect to on each specified host in your Hazelcast cluster.

Type: Integer

Required: No

Default: "5701"

auto_increment

Enable for Hazelcast to attempt to connect to the next incremental available port number if the configured port is unavailable.

Type: Boolean

Required: No

Default: "true"

hosts

List of hosts that can participate in the cluster.



Type: Array

Required: No

Default: ["localhost"],

man_center

Specifies the cluster information that you can view in the Hazelcast Management Center UI.

Type: List

Required: No

Default: {"enabled" : false, "host" : "localhost", "port" : 8091}

cluster_per_group

Enable the stateful information from each process group to persist in a dedicated Hazelcast cluster.

Type: Boolean

Required: No

Default: false

Moogfarmd Failover

keepalive_interval

Time interval in seconds at which processes report their active or passive status and check statuses of other processes.

Type: Integer

Required: No

Default: 5

margin

Amount of time in seconds after keepalive_interval before Cisco Crosswork Situation Manager considers processes that do not report their status to be dead.

Type: Integer

Required: No

Default: 10

failover_timeout

Number of seconds to wait for previously active process to become passive during a manual failover.



Type: Integer

Required: No

Default: 10

automatic_failover

Allow a passive process to automatically become active if no other active processes are detected in the same process group.

Type: Boolean

Required: No

Default: false

heartbeat_failover_after

Number of consecutive heartbeats that a process fails to send before Moogfarmd considers it inactive.

Type: Integer

Required: No

Default: 2

Process Monitor

heartbeat

Interval in milliseconds between heartbeats sent by processes.

Type: Integer

Required: Yes

Default: 10000

max_heartbeat_delay

Number of milliseconds to wait before declaring heartbeat as missing. Defaults to 10% of the heartbeat.

Type: Integer

Required: No

Default: 1000

Processes

Groups of processes that you want to be able to stop, start and restart from Self Monitoring in the Cisco Crosswork Situation Manager UI. For each group you can configure the following options:



group_name

Name of the process group that Cisco Crosswork Situation Manager uses when it starts and stops the service.

Type: String

Required: Yes

Default: N/A

instance

Name of the instance for the process.

Type: String

Required: Yes

Default: N/A

display_name

Additional identification label that appears in the UI.

Type: String

Required: No

Default: N/A

cluster

Name of the process's cluster. This overrides the default cluster for a process. If left empty, the Cisco Crosswork Situation Manager uses the process's default cluster.

Type: String

Required: No

Default: N/A

service_name

Name of the service script that Cisco Crosswork Situation Manager uses to control the process. If you do not configure a service name, Cisco Crosswork Situation Manager uses the group name, removing underscores and appending a 'd'. For example, "traplam" becomes "traplamd".

Type: String

Required: No

Default: N/A

process_type



Type of process. If left empty, Cisco Crosswork Situation Manager calculates the type based on the group name.

Type: String

Required: No

One of: moog_farmd, servlet, LAM

Default: N/A

reserved

Determines if the process produces a warning in the UI when it is running. Processes that are unreserved do not produce a warning.

Type: Boolean

Required: No

Default: true

subcomponents

Specifies which Moolets are reserved for the Moogfarmd process. If left empty, no Moolets are reserved for the Moogfarmd process.

Type: Array

Required: No

Default: N/A

Encryption

encryption_key_file

Default location of the encryption key file.

Type: String

Required: No

Default: "/location/of/.key"

High Availability (HA)

cluster

Default HA cluster name.

Type: String

Required: No

Default: "MOO"



Webhost

webhost

Default hostname that the LAMs use. Do not enter the port number or protocol here.

Type: String

Required: No

Default: "localhost"

Port Range

port_range_min

Minimum port number in the range that the Cisco Crosswork Situation Manager services use when they look for open ports:

Type: String

Required: No

Default: "50000"

port_range_max

Maximum port number in the range that the Cisco Crosswork Situation Manager services use when they look for open ports:

Type: String

Required: No

Default: "51000"

Configure Data Ingestion

Integrations and LAMs handle data ingestion from your event sources into Cisco Crosswork Situation Manager.

Many monitoring and ticketing systems can be configured by using an integration in the UI. Go to the **Integrations** tab to see what is available.

If you want to set properties that are not visible in the integration, or configure for high availability, modify the LAM configuration file instead. For each data source you can configure either the integration or the LAM, not both. A UI integration is independent from a LAM and you cannot edit it outside the UI.

You can find information about specific integrations and LAMs in the [Integrations Guide](#).Integrations



Custom Info

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.

Note

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

Note

Please note: This cannot be used with display only fields

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

-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 is 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
Note Please note: You can leave empty for all but this might take some time	
-l, --loglevel <arg>	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
<code>moog_remove_alert_custom_field</code>	This removes a Alert custom info field
<code>moog_remove_sitn_custom_field</code>	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 an 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 Parsing

Cisco Crosswork Situation Manager divides incoming data into tokens (tokenised) and then assembles the tokens into an event. You can control how tokenising works.

Start and End Characters

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.

Regular Expressions

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

```
parsing:
{
  type: "regexp",
    regexp:
    {
      pattern : "(?m)^START: (.*)$",
      capture_group: 1,
      tokeniser_type: "delimiters",
      delimiters:
      {
        ignoreQuotes: true,
        stripQuotes: true,
        ignores: "",
        delimiter: ["|", "\r"]
      }
    }
}
```

Delimiters

Delimiters define how string are split into tokens for processing. To process a comma-separated file, where a comma separates each value, define the comma as a delimiter.

Token are referenced from the start position starting at one (not zero).

For example, for the input string "the,cat,sat,on,the,mat" where the delimiter is a comma, token 1 is "the", token 2 "cat" and so on.

Combining tokenization and parsing can be complex. For example, if you use a comma delimiter and the token contains a comma, the token is split into two. To avoid this you can quote strings. You can then define whether to strip or ignore quotes.

An example delimiters section in a configuration file is as follows:

```
delimiters:
{
  ignoreQuotes      : true,
  stripQuotes       : false,
```



```
    ignores          : "",
    delimiter        : [",", "\"", "\r"]
}
```

When `ignoreQuotes` is set to true, all quotes are ignored and inputs are tokenised on the delimiters only.

When `ignoreQuotes` is false, delimiting does not occur until the matching end quote is found. This allows tokens to include delimiters. For example, given the following input when the delimiter is a comma:

hello world, "goodbye, cruel world".

Found tokens when `ignoreQuotes` is true: [hello world, goodbye, cruel world] (3).

Found tokens when `ignoreQuotes` is false: [hello world, "goodbye, cruel world"] (2).

Set `stripQuotes` to true to remove start and end quotes from tokens. For example, "hello world" results in a single token: [hello world].

`Ignore`s 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.

Mapping

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 },
  { name: "LastOccurrence", position: 1 }
]
```

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 the message bus, values from the tokens that are parsed. For example, in mapping there is a value called rules, which is a list of assignments.

```
mapping:
{
    catchAll: "overflow",
    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.

Using braces within mapping definitions allows you to include URLs and special characters. For example:

```
mapping:
{
    [
        { name: "type", rule: "${https://url}" },
        { name: "type", rule: "${https://url} customText" },
        { name: "type", rule: "${https://url}${keyA\\b\\c}" }
    ]
}
```

Escape backslashes (\\) and note that you cannot embed variables.



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 defined in `catchAll` and passed as part of the event.

Custom Info Mapping

You can define `custom_info` mapping in LAM configuration files. This allows you to configure a hierarchical structure. An example mapping configuration is:

```
mapping:
{
    rules:
    [
        { name: "custom_info.eventDetails.branch", rule: "$branch"
    },
        { name: "custom_info.eventDetails.location", rule:
"$location" },
        { name: "custom_info.ticketing.id", rule: "$incident_id" }
    ]
}
```

This produces the following `custom_info` structure:

```
"custom_info": {
    "eventDetails": {
        "branch": "Kingston",
        "location": "KT1 1LF"
    },
    "ticketing": {
        "id": 94111
    }
}
```

You can use braces within mapping definitions. This allows you to include URLs and special characters. For example:

```
{ name: "type", rule: "${https://url}" },
{ name: "type", rule: "${https://url} customText" },
{ name: "type", rule: "${https://url}${keyA.b.c}" }
```

Note that you must escape backslashes and you cannot embed variables.

Filtering

The `filter` defines whether a LAM uses a LAMbot. A LAMbot moves overflow properties to custom info and performs any actions that are configured in its LAMbot file. The LAMbot processing is defined in the `presend` property in the `filter` section of the LAM configuration file.

For example, the SolarWinds LAM configuration file contains this `filter` section:

```
filter:
{
```



```
modules : ["CommonUtils.js"],
presend : "SolarWindsLam.js"
}
```

This indicates that `SolarWindsLam.js` processes the events and then sends them to the Message Bus.

If you don't want to map overflow properties, you can comment out the `presend` property to bypass the LAMbot and send events straight to the Message Bus. This speeds up processing if you have a high volume of incoming alerts. Alternatively, you can define a custom stream to receive events. See [Alert Builder](#) for details.

See [LAMbot Configuration](#) for more information on the `presend` function.

The optional `modules` property can be used to provide a list of JavaScript files that are loaded into the context of the LAMbot and executed. It allows LAMs to share modules. For example, you can write a generic Syslog processing module that is used in both the Socket LAM and the Logfile LAM. This reduces the need for duplicated code in each LAMbot.

Conversion Rules

Conversion rules are used by Cisco Crosswork Situation Manager to convert received data into a usable format, including severity levels and timestamps.

Severity

The following example looks up the value of severity and returns the mapped integer.

```
conversions:
{
  sevConverter:
  {
    lookup  : "severity",
    input   : "STRING",
    output  : "INTEGER"
  },
},
constants:
{
  severity:
  {
    "CLEAR"                : 0,
    "INDETERMINATE"        : 1,
    "WARNING"              : 2,
    "MINOR"                : 3,
    "MAJOR"                : 4,
    "CRITICAL"             : 5,
    moog_lookup_default    : 3
  }
}
```



In the above example:

- conversions receives a text value for severity.
- sevConverter uses a lookup table "severity" to reference a table named severity defined in the constants section.
- The integer value matching the text value is returned.
- moog_lookup_default is used to specify a default value when a received event does not map to a listed value.

For example, the text value "MINOR" is received and the integer value 3 is returned.

If moog_lookup_default is not used and a received event severity does not map to a specifically listed value, the event is not processed.

See [Severity Reference](#) for more information about the severity levels in Cisco Crosswork Situation Manager.

Time

Time conversion in Cisco Crosswork Situation Manager supports the Java platform standard API specification. See [Simple Date Format](#) for more information.

Some Unix time formats are indirectly supported and LAM logging indicates any automatic conversion that occurred at startup.

The only PCRE/Perl modifier automatically converted is the lone 'U' ungreedy modifier, PCRE's '-U' is not supported. If the pattern contains a -U it should be removed manually.

You can specify a time zone configuration so the LAM parses the incoming timestamps with the expected time zone. For example:

```
conversions:
{
    timeUnitConverter:
    {
        timeUnit      : "MILLISECONDS",
        input          : "STRING",
        output         : "INTEGER"
    },
    timeConverter:
    {
        timeFormat     : "%Y-%m-%dT%H:%M:%S",
        timeZone       : "UTC",
        input          : "STRING",
        output         : "INTEGER"
    }
}
```

You can specify the timezone name or abbreviation. See [List of TZ Database Time Zones](#) for the full list.



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 `xNode="acmeSvr01"`, `$Severity="Major"` and so on.

Severity Reference

Severity is a measure of the seriousness of an event and indicates how urgently it requires corrective action.

Cisco Crosswork Situation Manager LAMs and integrations use six industry standard severity levels as follows:

- 0: Clear - One or more events have been reported but then subsequently cleared, either manually or automatically.
- 1: Indeterminate - The severity level could not be determined.
- 2: Warning - A number of faults with the potential to affect services have been detected.
- 3: Minor - A fault that is not affecting services has been detected. Action may be required to prevent it from becoming a more serious issue.
- 4: Major - A fault is affecting services and corrective action is required urgently.
- 5: Critical - A serious fault is affecting services and corrective action is required immediately.

The severity mapping is set in each LAM configuration file:

```
severity:  
{
```



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

The LAM takes the severity string in a received event and translates it into one of the above integer values using the mapping in its configuration file:

```
sevConverter:
{
    lookup  : "severity",
    input   : "STRING",
    output  : "INTEGER"
},
mapping:
    rules:
    [
        { name: "severity", rule:
"$severity",conversion:"sevConverter"},
    ]
```

You can customize the severity section of the LAM configuration file according to the severities used in the system sending events to Cisco Crosswork Situation Manager. In the following example, events sent to the LAM with non-standard severities 'info' and 'Information' are mapped to 'INDETERMINATE' in Cisco Crosswork Situation Manager:

```
severity:
{
    "info"                : 1,
    "Information"         : 1,
    "user"               : 1,
    "warning"            : 2,
    "Warning"            : 2,
    "error"              : 5,
    moog_lookup_default  : 1
}
```

The `moog_lookup_default` property specifies a default value to use when the severity does not match any of the defined strings. If you do not set a default, events with an unmapped severity are not processed. For more information on mapping see "Conversion Rules" in [Data Parsing](#).

Cisco Crosswork Situation Manager determines a Situation's severity from the member alert with the highest severity level.

Configure Data Processing

Moogfarmd is the core system application that runs all of the algorithms and automation relevant to Cisco Crosswork Situation Manager. It is responsible for the following:



- Creating alerts.
- Analyzing alerts to determine their significance.
- Clustering alerts into Situations.
- Performing automation relating to the automated response such as escalation, routing, notification, invitation of either alerts or Situations.

The topics in this guide help you configure the data processing components of Moogfarmd:

You can run one or many instances of Moogfarmd on your Cisco Crosswork Situation Manager system.

Services

The Cisco Crosswork Situation Manager installation installs Moogfarmd as a service:

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

A backup Moogfarmd service script is located at `$MOOGSOFT_HOME/etc/service-wrappers/moogfarmd`.

If you run multiple instances of Moogfarmd on the same host, copy and modify the default Moogfarmd service script for each Moogfarmd running on the host:

1. Copy `$MOOGSOFT_HOME/etc/service-wrappers/moogfarmd` to `/etc/init.d/mymoogfarmd`.
2. Edit the following parameters in the `/etc/init.d/mymoogfarmd` file:

```
SERVICE_NAME=mymoogfarmd  
CONFIG_FILE=$PROCESS_HOME/config/my_moog_farmd.conf
```
3. You now have a new service to be used to start your own specific Moogfarmd:

```
service mymoogfarmd start
```

For information on starting, stopping and configuring Moogfarmd, see the [Moogfarmd Reference](#).

Alert Processing

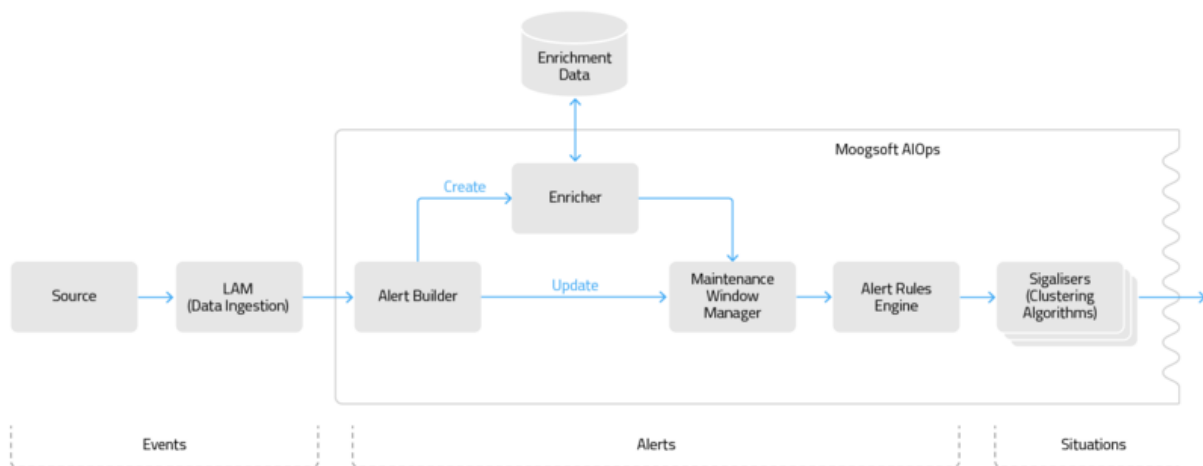
Cisco Crosswork Situation Manager processes alerts using the following components. These components are responsible for performing analysis, adding information to alerts, and noise reduction techniques.

- [Events Analyser](#): A standalone process that analyses tokens in events and assigns each token an [entropy](#) value. The Events Analyser can use any text field in an event but, by default, it uses the event's description. This process runs periodically and does not form a part of the alert processing workflow.



- **Alert Builder:** Processes events from the Message Bus. It:
 - Deduplicates events into alerts.
 - Calculates the entropy of alerts.
- **Enricher:** Enriches alerts with additional information.
- **Maintenance Window Manager:** Marks alerts as 'In maintenance' if they match a scheduled maintenance window filter. You can set up maintenance windows for planned maintenance, such as scheduling a fix or regular maintenance of a system.
- **Alert Rules Engine:** Allows conditional processing of alerts, such as managing link up/link down processing.
- **Empty Moolet:** An optional component that enables further processing of alerts or Situations. It usually runs as a standalone process but it can also be embedded in the processing chain. Cisco Crosswork Situation Manager provides an example Empty Moolet in the form of an **Alert Manager**.

The following diagram shows the alert processing components in a typical implementation of a workflow chain in Cisco Crosswork Situation Manager:



Each component comprises a Moolet supplemented by Moobots.

Events Analyser

- **Stream and Partition-based Analysis**
- **Natural Language Processing Analysis**
 - Tokenization of Text
 - Token Type Identification
 - Token Masking
 - Language Processing Techniques



- [Priority Words](#)
- [Token Variation Threshold](#)

The Events Analyser utility is a standalone process. It uses [Natural Language Processing \(NLP\)](#) techniques to analyze inbound event data. The Events Analyser divides text fields within the events into [tokens](#). Based on the frequency of these tokens appearing in other events, it assigns an entropy value to the tokens and to the alerts in Cisco Crosswork Situation Manager. See [Entropy Overview](#) for more information on how Cisco Crosswork Situation Manager evaluates entropy and uses entropy thresholds to reduce the level of 'noise' from incoming event data.

Stream and Partition-based Analysis

You can configure Cisco Crosswork Situation Manager so that the Events Analyser calculates the entropy values for events from different streams for Cisco Crosswork Situation Manager as a whole, even though those streams have no relationship with each other.

You can also configure the Events Analyser so that it calculates the entropy values for events for different partitions. As an example, you may want to run separate entropy calculations for different regions. In this case, you should specify the alert field that identifies the region in the `partition_by` field in the [Events Analyser configuration file](#). In this type of configuration, the same token can be given multiple entropy values within the same Moogfarmd deployment based on its frequency in the events within each partition. You can set up different configuration options for the different partitions. For example, in a particular partition, IP addresses may be masked whilst for another partition that may be unnecessary. In general, if a deployment uses the “pre-partition” method in Moogfarmd, that deployment benefits from partition-based entropy calculations.

See [Multiple Streams and Partitions](#) for more information on running the Events Analyser with different streams and partitions. See [Configure Events Analyser](#) for further information on non-partitioned and partitioned configurations.

Natural Language Processing Analysis

The Events Analyser utility performs a number of linguistic analyses on events. It then uses this linguistic analysis to calculate an entropy value for each token and then for every alert.

Tokenization of Text

The Events Analyser splits a text string at word boundaries, such as spaces or punctuation marks, into blocks. Each block of text is known as a token. For example, the following description has five tokens:

Link down on port 2/32

Token Type Identification



Commonly used word boundaries are often integral to the meaning of a token, for example, dots in IPV4 addresses. The Events Analyser identifies complete tokens of the following types within the structure of an event:

- IP addresses:
 - v4
 - v6
- MAC addresses
- OIDs
- Dates
 - Most standard formats
- Numbers:
 - Integers
 - Real numbers
 - With and without unit suffixes, for example, 99%, 12kb, 345ms
- File paths:
 - Forward slashes
 - Backward slashes
- GUIDs
- Hexadecimal numbers:
 - With and without the 0x prefix
- URLs
- Email addresses:
 - Most standard formats

Identifying token types in arbitrary text is not an exact science and so, occasionally, the algorithms may identify tokens as a certain type which seems incorrect to a human.

After the Events Analyser has identified the token types, it can use them for masking and to identify tokens with high variation in a given alert.

Token Masking

Tokens that change between events for the same alert can cause that alert to be assigned an incorrectly high entropy value. The most obvious example involves dates and times. If



the description of an event is to be analyzed but each event contains a different timestamp, that timestamp will have a high entropy and skew the entropy for that alert as a whole. For other token types that change frequently, such as URLs or IP addresses, it may be desirable to retain the higher entropy associated with that token type because the changing value is significant.

You can configure the Events Analyser to include or exclude specific token types in the entropy analysis for each event partition.

You should consider masking dates, times and numbers from the entropy calculation.

Language Processing Techniques

The Events Analyser uses many standard techniques in language processing:

- **Case Folding**
 - Tokens that differ only by case, for example, 'WORD', 'Word' or 'word', are converted to the same case and considered equal.
 - Case folding is applied to all token types.
- **Stop Words**
 - You can add common or meaningless words, such as 'a', 'be', 'not', to a stop words file so that they are removed from the entropy calculation.
 - You can define a universal 'length' parameter so that any word below a certain length is treated as a stop word. For example, if set to '2', any words of one or two characters are ignored.
 - Stop words are applied to all token types.
- **Stemming**
 - A technique used to reduce a word to its root to remove plurals or different tenses in verbs. Words with the same root are considered equal.
 - Note that some words, when stemmed, look unusual. For example, 'priority', 'priorities', prioritize, get stemmed to 'priorit'.
 - If stemming is enabled, the stemmed form is stored in the reference database.
 - Stemming is only applied to tokens of type 'word', that is, it is not applied to numbers, GUIDs, IP addresses, etc.

Priority Words

Priority words are similar in concept to stop words but, rather than removing that word from the analysis as occurs with stop words, a priority word is assigned an entropy value of 1. For example, if 'reboot' is defined as a priority word, any tokens containing



the word 'reboot' are given an entropy value of 1 regardless of how frequently the word appears in events.

Note

- Priority words are analyzed after stop words. If a token satisfies the criteria of a stop word, it is removed from the analysis and so cannot subsequently be considered as a priority word.
- The reference database contains the calculated entropies for all tokens regardless of whether they are classed as priority words.

Token Variation Threshold

Token variation threshold analysis involves the different forms of each field and how the tokens in those different forms vary between events in the same alert. This is most easily explained by an example. Assume that all token masking is off and that an alert consists of the following six events:

QDepth beyond 90% threshold on host = 22222

QDepth beyond 90% threshold on host = 44444

QDepth beyond 90% threshold on host = 44444

QDepth beyond 90% threshold on host = 11111

QDepth beyond 90% threshold on host = 44444

The value for the host is changing between events, there are three occurrences of 44444 and one occurrence of each of the other values. Values that appear infrequently can skew the entropy value for the alert. In order to prevent this skewing, you can apply a threshold. The threshold is a ratio between 0 and 1, where 0 implies that a token can appear only once and still contributes to the entropy calculation, while a value of 1 implies that the value must be the same in every event before it is considered. If a threshold of 0.5 were used, the value 44444 would contribute to the entropy, but the values 11111 and 22222 would not, because only the value 44444 appears in half of the events in the alert.

The Events Analyser performs this analysis for each form of each field within each event of every alert.

Entropy Overview

Entropy is defined as the degree of disorder or randomness in a system. In Cisco Crosswork Situation Manager, entropy is a measure of how unexpected or unpredictable an event or an alert is. According to information theory, the more unpredictable or unexpected an event is, the more information it is deemed to carry. Therefore, entropy is a measure of the amount of information contained in an event.

The [Events Analyser](#) utility is a standalone process that assigns an entropy value to an event token based on its uniqueness. The [Alert Builder](#) assigns an entropy value to each alert based on the token entropies. The entropy value is a numeric value between 0 and



1 (accurate to 16 decimal places). It provides an indication of how important an alert is. An entropy value of 0 means that the alert is just 'noise' and a value of 1 means that the alert is significant. You can configure the Sigaliser [clustering algorithms](#) to ignore common alerts with a low entropy value; this reduces 'noise' in Cisco Crosswork Situation Manager. Clustering Algorithm Guide

How Cisco Crosswork Situation Manager Evaluates Entropy

The Events Analyser utility analyzes the text attributes of events to assign a semantic entropy value. In the default Cisco Crosswork Situation Manager implementation, the Events Analyser uses the description field but you can configure it to use other text fields. The Events Analyser divides the text in between spaces into tokens. For example, the following description has five tokens:

Link down on port 2/32

The Events Analyser calculates the entropy of each token and stores the token in the Cisco Crosswork Situation Manager reference database with its associated entropy value. Initially, a new token has a value of 1. The Events Analyser reduces this entropy value as more events occur which contain the same token.

You can configure the Events Analyser to mask volatile token types, such as dates, times, numbers, URLs or IP addresses, so that they are not included in the tokens. See [Events Analyser](#) for further details of the analysis it performs.

The Alert Builder uses the entropy value of the tokens within an alert to calculate the entropy of that alert.

The Events Analyser uses the EntropyV2 calculation method in the default Cisco Crosswork Situation Manager implementation. The EntropyV2 method calculates entropy values in real-time based on any tokens it has encountered before. The Alert Builder assigns the entropy of an alert based on the entropy value of the tokens within the alert rather than the entire database. Tokens within an alert which occur frequently contribute negatively to the entropy of an alert, indicating that the alert may not be as significant as an alert with tokens that are seen less frequently. This is in contrast to the EntropyClassic algorithm where the entropy of each alert takes into consideration the significance of tokens in the entire database.

Note

Cisco recommends using the EntropyV2 algorithm to produce better alert entropy values than with the EntropyClassic algorithm.

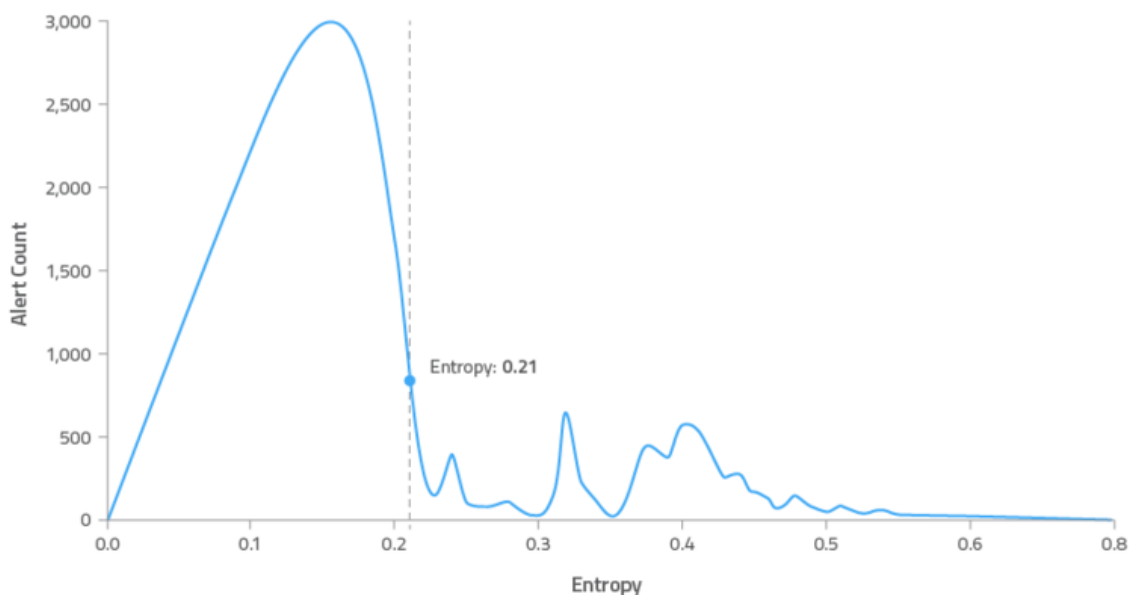
If the Alert Builder receives an event with a token that it has encountered before, from a previous run of the Events Analyser, it sets the alert entropy to match the value saved in the reference database. If the Alert Builder receives an event with a token that it has not encountered before, it calculates the entropy value in real-time and applies this value to the alert. The Alert Builder also saves the entropy value in the reference database for future retrieval.



The Events Analyser stores data in memory while it calculates entropy values. It is important that the Events Analyser runs frequently to ensure that it does not fail with a memory outage. See [Run Events Analyser](#) for more information on running the Events Analyser.

Set an Entropy Threshold

You can set an entropy threshold in each Sigaliser so that only alerts with a higher entropy value are included in Situations. To decide on the value of your entropy threshold, consider the distribution of entropy values in the alerts. A typical entropy value distribution is shown in the following diagram:



Cisco recommends that you set your entropy threshold to a value on the downward slope of the peak to exclude the majority of alerts. In this example, the entropy threshold is set at 0.21. This reduces the level of 'noise' so that you are only clustering the important alerts, with an entropy value greater than the threshold, into Situations.

You can define entropy thresholds in the [Sigalisers](#) to exclude alerts which have an entropy value that is lower than the threshold. This prevents Cisco Crosswork Situation Manager from including unimportant 'noisy' alerts in Situations.

Vertex Entropy

[Vertex Entropy](#) uses a different form of entropy, topological entropy, to establish how critical the nodes are in your network topology. You can use Vertex Entropy calculations within Cookbook to create Situations which cluster alerts from important nodes.



Configure Events Analyser

You can configure the Events Analyser to analyze all the event data received by Cisco Crosswork Situation Manager together or to analyze event data by partitions or streams. See the [Events Analyser](#) for more information on these options.

- [Example of Non-Partitioned Data](#)
- [Example of Partitioned Data](#)
- [Disabling Entropy Calculations](#)

Edit the configuration file at `$MOOGSOFT_HOME/config/events_analyser.conf` to control the behavior of the Events Analyser.

See the [Events Analyser Reference](#) for a full description of all properties. Some properties in the file are commented out by default. Uncomment the properties to enable them.

To configure the Events Analyser:

1. The default configuration uses the EntropyV2 calculation method. Cisco recommends using the default EntropyV2 calculation method for calculating entropy values because it has improved modelling of alert probabilities. However, if you want to, you can change the setting to use the EntropyClassic calculation method. Entropy data for EntropyClassic and EntropyV2 calculation methods are not compatible. If you switch between the two calculation methods, you must execute a full priming run of the Events Analyser after you have changed the setting to ensure that all the entropy data matches the same configuration. See [Run Events Analyser](#) for further details on executing a full priming run of the Events Analyser.
2. Use the default values for the `priming_source_data`.
3. Configure whether or not the Events Analyser partitions the entropy data. See the examples of [non-partitioned data](#) and [partitioned data](#) for further details.
4. Configure the "default" Events Analyser behavior. See the example of [non-partitioned data](#) for further details.
5. If using partitioned data, configure the Events Analyser for any partitions that you want to behave differently. If you do not add a separate configuration for a partition, the Events Analyser uses the "default" configuration for that partition. The Events Analyser also uses the "default" configuration for any properties that are not defined in a partition configuration. See the example of [partitioned data](#) for further details.

Example of Non-Partitioned Data

The default configuration file at `$MOOGSOFT_HOME/config/events_analyser.conf`, similar to the example shown below, contains a non-partitioned configuration. The "partition_by" property has been set to null to show that the entropy data is not to be



partitioned. The "default" settings have been configured for all entropy values. See the [Events Analyser Reference](#) for further information on these properties.

```
{
  "entropy_calc": "EntropyV2",
  "priming_source_data" :
    {
      "alerts_table" : "alerts",
      "events_table" : "events",
      "snapshots_table" : "snapshots",
      "timestamp_column" : "last_event_time"
    },
  "partition_by" : null,
  "default" :
    {
      "fields" :
        [
          "description"
        ],
      "mask" :
        {
          "ip_address"      : false,
          "mac_address"     : false,
          "oid"              : false,
          "date_time"       : true,
          "number"          : true,
          "path"             : false,
          "guid"             : false,
          "hex"              : false,
          "url"              : false,
          "email"            : false,
          "word"             : false,
          "stop_word"       : false
        },
      "casefold" : true,
      "stop_words" : true,
      "stop_word_length" : 0,
      "stop_word_file" : "stopwords",
      "priority_words" : false,
      "priority_word_file" : "prioritywords",
      "stemming" : false,
      "stemming_language" : "english",
      "token_variation_threshold" : 0.5
    }
}
```

Example of Partitioned Data

The example below shows additional configuration of the Events Analyser for two partitions "san_francisco" and "new_york". These settings override the "default" configuration in the example of [non-partitioned data](#) above.

In this example, the source field is used to partition the entropy data:



```
"partition_by" : "source",
```

The configuration for the "san_francisco" partition uses the description, agent and source fields for calculating entropy values and does not use stop words. The "new_york" partition uses different masking properties to the "default" configuration: date_time is not masked but ip_address, email, and url are masked. This partition also uses stemming for calculating entropy values. Since the language is not specified, the default of English is used. All other properties that have not been configured in these partitions will use the properties in the "default" configuration.

If there are any other partitions, for example, "los_angeles", that do not have any properties specified in the configuration file, they will use the "default" configuration.

See the [Events Analyser Reference](#) for further information on these properties.

```
, "partition_overrides" :
{
  "san_francisco" :
  {
    "fields" :
    [
      "description", "agent", "source",
    ],
    "stop_words" : false
  },
  "new_york" :
  {
    "mask" :
    {
      "date_time" : false,
      "ip_address" : true,
      "email" : true,
      "url" : true
    },
    "stemming" : true
  }
}
```

Disabling Entropy Calculations

Cisco recommends that you configure the [Sigaliser clustering algorithms](#) to use entropy thresholds so that they exclude 'noisy' alerts which contain low levels of important information. This allows operators to concentrate on Situations containing important alerts. However, if you do not intend to use entropy calculations, you should: Clustering Algorithm Guide

- Set the 'entropy_calc' property to 'EntropyClassic'.
- Set the 'properties_from_db' property to 'false' for all running Alert Builder Moolets.



Run Events Analyser

The Events Analyser is responsible for analysing the tokens within alerts and calculating their entropy values. The Events Analyser updates the alerts with the calculated entropy value and also updates the reference database with all the tokens and their associated entropy values.

- [Command Line Options](#)
- [Run Events Analyser](#)
 - [Daily Run](#)
 - [Hourly Run](#)
 - [Run Events Analyser Manually](#)
- [Multiple Streams and Partitions](#)
- [Usage Examples](#)

Command Line Options

The `events_analyser` command line executable accepts the following options:

Option	Input	Description
<code>--config <arg></code>	String: <file path/name>	Name and path of the configuration file specific to running the Events Analyser. The default is <code>events_analyser.conf</code> . Example: <code>--config=\$MOOGSOFT_HOME/etc/events_analyser.conf</code>
<code>-l, --loglevel <arg></code>	One of: ALL, INFO, WARN, NONE	Specifies the amount of logging information. Defaults to WARN, which is the recommended level in all production implementations.
<code>--incremental</code>	-	Analyzes only new event data that was received since the last time the Events Analyser was run.
<code>--readage <arg></code>	Number, followed by one of: <ul style="list-style-type: none">• s (seconds)• m (minutes)	Amount of data to analyse, in seconds, minutes, hours, days or weeks. Example: <code>--readage 2w</code>



	<ul style="list-style-type: none">• h (hours)• d (days)• w (weeks)	
--keepage <arg>	Number, followed by one of: <ul style="list-style-type: none">• s (second s)• m (minute s)• h (hours)• d (days)• w (weeks)	Amount of data to keep, in seconds, minutes, hours, days or weeks. Example: --keepage 30d
--stream <arg>	String: <alert stream name>	Stream name to be given to the current analysis. Example: --stream "PRIMARY"
--partition <arg>	String: <partition value>	Name of the partition to be analyzed. It must be a valid value of the partition_by field. Example: --partition "SanFrancisco"

Run Events Analyser

Cisco recommends that you run the Events Analyser regularly as follows:

- **Daily:** analyzes the last two weeks of data.
- **Hourly:** in incremental mode which analyses all new event data since the last time the Events Analyser was run.

These default settings are specified in `moog_init_server.sh`.

You can also run the Events Analyser manually on an ad hoc basis.

Daily Run



To initiate a daily run, that is, where all entropy values are calculated for the last two weeks of event data, you should specify the Events Analyser to run with the following command line options:

```
./events_analyser --readage 2w
```

In this case, the Events Analyser:

- Uses the default configuration file `$MOOGSOFT_HOME/etc/events_analyser.conf`.
- Analyzes all data received in the last two weeks, based on the `timestamp_column` property in the `events_analyser.conf` file.
- Adds all analyzed data to the reference database for the default stream.
- Leaves any data for other, named streams unchanged.

Hourly Run

The Events Analyser utility provides the ability for incremental priming. When the Events Analyser utility is run repeatedly with the `--incremental` option, each subsequent run of the utility analyses the event data starting from the last analyzed event. For example, if the first run analyzes data up to event ID = 666, the next incremental run of the utility analyzes data from 667 to say 999, the third incremental run reads in data from event ID 1000, and so on.

To initiate an hourly run, that is, where all entropy values are calculated since the last analyzed event, you should specify the Events Analyser to run with the following command line options:

```
./events_analyser --incremental
```

In this case, the Events Analyser:

- Uses the default configuration file `$MOOGSOFT_HOME/etc/events_analyser.conf`.
- Analyzes all data since the last incremental run, based on the `timestamp_column` property in the `events_analyser.conf` file.
- Adds all analyzed data to the reference database for the default stream.
- Leaves any data for other, named, event streams unchanged.

Run Events Analyser Manually

To run the Events Analyser manually, you can run it without any command line options. This command runs the Events Analyser for all new event data received in the last two weeks or since the last analysis, whichever is most recent.

```
./events_analyser
```

In this case, the Events Analyser:

- Uses the default configuration file `$MOOGSOFT_HOME/etc/events_analyser.conf`.



- Analyzes all event data received in the last two weeks or since the last time the Events Analyser was run, whichever is most recent, based on the `timestamp_column` property in the `event_analyser.conf` file.
- Adds all analyzed data to the reference database for the default stream.
- Leaves any data for other, named, event streams unchanged.

To run the Events Analyser to analyze event data over a longer period, you should include the `--readage` option. In this example, the `--readage` option is set to 13 weeks:

```
./events_analyser --readage 13w
```

In this case, the Events Analyser:

- Uses the default configuration file `$MOOGSOFT_HOME/etc/events_analyser.conf`.
- Analyzes all event data received in the last 13 weeks.
- Adds all analyzed data to the reference database for the default stream.
- Leaves any data for other, named, event streams unchanged.

Note

If you use a large value in the `--readage` option, you may find that the Events Analyser fails to complete the analysis. If this occurs, rerun it using a shorter period of time.

Multiple Streams and Partitions

You can run the Events Analyser for specific streams or partitions. In this example, the `--stream` option is specified to add the analyzed data to the "SECONDARY" event stream. The `--readage` option restricts the data analyzed to the last eight weeks of event data.

```
./events_analyser --stream "SECONDARY" --readage 8w
```

In this case, the Events Analyser:

- Uses the default config file `$MOOGSOFT_HOME/etc/events_analyser.conf`.
- Analyzes all event data received in the last eight weeks, based on the `timestamp_column` property in the `event_analyser.conf` file.
- Adds all analyzed data to the reference database for the "SECONDARY" event stream.
- Leaves data for all other, named, event streams unchanged.

You can use the `--partition` option to limit the data that is analysed to a specified partition. In this example, the `--readage` option restricts the data analyzed to the last four weeks of event data:



```
./events_analyser --stream "SECONDARY" --partition "SanFrancisco" --readage 4w
```

In this case, the Events Analyser:

- Uses the default config file `$MOOGSOFT_HOME/etc/events_analyser.conf`.
- Analyzes all event data received in the last four weeks for the "SanFrancisco" partition only.
- Adds all analyzed data to the reference database for the "SanFrancisco" partition in the "SECONDARY" event stream.
- Leaves data for all other event streams and partitions unchanged.

Note

Cisco recommends that you always use the `--readage` option when analyzing streams or partitions to ensure that the Events Analyser processes the required amount of data. If the `--readage` option is not specified, the Events Analyser only analyzes new event data received in the last two weeks or since the last analysis, whichever is the most recent, regardless of whether this was for a different stream or partition.

Usage Examples

There are many combinations of command line options. Some common usage scenarios include:

Command Line Options	Typical Use Case
<none>	<ul style="list-style-type: none">• Events analysis to be run incrementally.• Uses the default configuration for all new event data in the last two weeks or since the last analysis, whichever is most recent.• Updates the reference database with the new data for the default stream.
<code>--readage 4w</code>	<ul style="list-style-type: none">• Events analysis to be run nightly.• Uses the default configuration for the last four weeks of event data.• Updates the reference database with the new data for the default stream.
<code>--incremental</code>	<ul style="list-style-type: none">• An incremental events analysis to be run hourly.• Uses the default configuration for the all new event data since the last run.• Updates the reference database with the new data for the



default stream.

- | | |
|----------------------|--|
| --incremental | • An incremental events analysis to be run hourly. |
| --keepage 2w | • Uses the default configuration for all new event data received since the last run. |
| | • Removes all data from the reference database for the default stream that is more than two weeks old. |
| --stream "PRIMARY" | • Performs an events analysis analyzing only those events in the "London" partition. |
| --partition "London" | |
| --readage 13w | • The data is written to the "PRIMARY" event stream. |
| | • Data for all other streams remains unchanged. |
| | • Data for all other partitions in the "PRIMARY" stream remains unchanged. |

Events Analyser Reference

This is a reference for the [Events Analyser](#) utility. The Events Analyser configuration properties are found in `$MOOGSOFT_HOME/config/events_analyser.conf`.

entropy_calc

Entropy calculation method. Cisco recommends using the EntropyV2 calculation method for more accurate entropy values.

Type: String

Required: Yes

One of: EntropyV2, EntropyClassic

Default: "EntropyV2"

priming_source_data

Source data to use when priming the entropy value database table, that is, running the Events Analyser to calculate entropy values. By default, the priming source data is taken from tables in the main database schema called moogdb. `timestamp_column` is a column in the `snapshots_table`.

Type: String

Required: Yes

Default:

```
{
    "alerts_table" : "alerts",
```



```
"events_table" : "events",  
"snapshots_table" : "snapshots",  
"timestamp_column" : "last_event_time"  
}
```

partition_by

Identifies the properties in each event that is used to partition them so that they are grouped separately by the Sigalisers. If partitioning is enabled, the following properties can be configured independently for each partition. See [Configure Events Analyser](#) for further details on partitions and configuration examples.

Type: String

Required: Yes

Default: null

Example: "partition_by" : "source"

fields

Properties in each event that contribute to the entropy value calculation.

Type: List of strings

Required: Yes

Default: "description"

mask

Token types to be included or excluded from entropy calculations. If a token type is set to false, the entropy calculation includes it. If it is set to true, the entropy calculation excludes the token type. Masking token types, such as dates or numbers, ensures that tokens are not given a higher entropy value than they should have because of unique numbers or dates.

Type: Boolean

Required: No

Default:

```
{  
  "ip_address" : false,  
  "mac_address" : false,  
  "oid" : false,  
  "date_time" : true,  
  "number" : true,  
  "path" : false,  
  "number" : false,  
  "path" : false,  
  "guid" : false,  
}
```



```
"hex" : false,  
"url" : false,  
"email" : false,  
"word" : false,  
"stop_word" : false  
}
```

casefold

Whether tokens that differ only by case should be considered the same in entropy calculations.

Type: String

Required: Yes

Default: true

stop_words

Whether specific tokens should be ignored in entropy calculations. Stop words are small common words such as 'about', 'at' or 'the'.

Type: String

Required: Yes

Default: true

stop_word_length

Any token of this length or shorter is considered a stop word and is excluded from entropy calculations. The default of 0 means that no words are considered as stop words.

Type: Number

Required: Yes

Default: 0

stop_word_file

Path (optional) and name of the file containing a list of stop words to be excluded from entropy calculations. If you provide a file name only, the Events Analyser assumes the path \$MOOGSOFT_HOME/config/. The Events Analyser uses the full path if you provide it. The default Cisco Crosswork Situation Manager implementation provides a file named stopwords in \$MOOGSOFT_HOME/config/, which contains a list of common stop words.

Type: String

Required: Yes

Default: "stopwords"



priority_words

Whether priority words are included in entropy calculations. Alerts containing priority words are automatically given a maximum entropy value of 1.

Type: String

Required: Yes

Default: false

priority_word_file

Path (optional) and name of the file containing a list of stop words to be excluded from entropy calculations. If you provide a file name only, the Events Analyser assumes the path `$MOOGSOFT_HOME/config/`. The Events Analyser uses the full path if you provide it. The file `prioritywords` in `$MOOGSOFT_HOME/config/` is empty in the default Cisco Crosswork Situation Manager implementation.

Type: String

Required: Yes

Default: "prioritywords"

stemming

Whether words with the same word stem are to be considered as the same word in entropy calculations. For example, should 'fail', 'failed' and 'failing' all be considered as the same word.

Type: String

Required: Yes

Default: false

stemming_language

Language used in the events.

Type: String

Required: Yes

Default: "english"

Alert Builder

- [Configure Alert Builder](#)
- [Example Configuration](#)
- [Alert Builder Moobot](#)



The Alert Builder Moolet assembles alerts from incoming events, sent by the LAMs across the Message Bus. These alerts 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.
- 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 Cisco Crosswork Situation Manager.

Configure Alert Builder

Edit the configuration file at `$MOOGSOFT_HOME/config/moolets/alert_builder.conf`.

See [Alert Builder Reference](#) for a full description of all properties. Some properties in the file are commented out by default.

Example Configuration

The following example demonstrates a simple Alert Builder configuration:

```
{
    name                : "AlertBuilder",
    classname            : "CAAlertBuilder",
    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
}
```

Alert Builder Moobot

The Moobot, `AlertBuilder.js`, is associated with the Alert Builder Moolet. It undertakes most of the activity of the Alert Builder. When the Alert Builder Moolet processes an event, it calls the JavaScript function, `newEvent`:

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

The function `newEvent` contains a call to create an alert. The newly created alert is broadcast on the Message Bus.

See [Moobot Modules](#) for further information about Moobots. Moobot Modules

[Alert Builder Reference](#)

This is a reference for the [Alert Builder](#) Moolet.



You can change the behavior of the Alert Builder by editing the configuration properties in the `$MOOGSOFT_HOME/config/moolets/alert_builder.conf` configuration file. It contains the following properties:

name

Name of the Alert Builder Moolet. Do not change.

Type: String

Required: Yes

Default: "AlertBuilder"

classname

Moolet class name. Do not change.

Type: String

Required: Yes

Default: "CAAlertBuilder"

run_on_startup

Determines whether the Alert Builder runs when Cisco Crosswork Situation Manager starts. By default, it is set to true, so that when Moogfarmd starts, it automatically creates an instance of the Alert Builder. In this case you can stop it using `farmd_ctrl`.

Type: Boolean

Required: Yes

Default: true

moobot

Specifies a JavaScript file found in `$MOOGSOFT_HOME/moobots`, which defines the Alert Builder Moobot, which creates alerts.

Type: String

Required: Yes

Default: AlertBuilder.js

metric_path_moolet

Determines whether or not Cisco Crosswork Situation Manager includes the Alert Builder in the Event Processing metric for [Self Monitoring](#). Self Monitoring

Type: Boolean

Required: Yes



Default: true

event_streams

A list of event streams, which the Alert Builder Moolet processes in this instance of Moogfarmd. The LAMs can be configured to send events on different streams. Moogfarmd, as specified in the Alert Builder configuration, then decides whether or not to process them. If Cisco Crosswork Situation Manager runs multiple Moogfarmds, you can have different event streams being processed by different Alert Builder Moolets.

You can comment out event_streams, or provide an empty list. Then, the Alert Builder processes every event that is published on the default /Events topic on the Message Bus.

You configure the Alert Builder Moolet by giving it a list of strings, for example, ["App A", "App B"]. The result is that the Alert Builder listens for events published on /Events/AppA, and /Events/AppB, and processes that data. Importantly, in this example, events published to /Events or any other stream are ignored. You can have Moogfarmds that process completely separate event streams, or, multiple Moogfarmds that process some different event streams and some common event streams. You would do this when some of the alerts are common to all the applications that are being processed, but some are specific only to a given application. In this way, you can cluster alerts separately for each application by configuring the Sigalisers to only processes alerts from a specific upstream Alert Builder Moolet.

For example, if you have two separate applications that share the same network infrastructure: in Moogfarmd 1, you can have as the event streams, application A and networks, and, in Moogfarmd 2, you can have application B and networks. With this configuration, you can detect alerts and then create Situations that are relevant for just application A and similarly just for application B; however, if there is common networking infrastructure and problems occur with network failures across applications A and B, the Alert Builder can cluster these into Situations.

Type: String

Required: No

Default: ["AppA"]

threads

Specifies the number of threads in the Alert Builder. Choose a value to match the event rate experienced by your system that allows time for alert creation.

Type: String

Required: Yes

Default: 4

events_analyser_config



Allows you to specify a different Events Analyser configuration, for tokenizing and analysis rules, for each Alert Builder Moolet. If no configuration file is specified, the system default `events_analyser.conf` is used.

Type: String

Required: No

Default: "events_analyser.conf"

priming_stream_name

Stream name under which the Events Analyser runs in order to calculate token and alert entropies. If set to `null`, all alerts from all streams are included in the entropy calculations.

Type: String

Required: Yes

Default: `null`

priming_stream_from_topic

If set to `true`, Moogfarmd extracts the priming stream name from the event's stream. If set to `false`, Moogfarmd uses the stream configured in `priming_stream_name`.

Type: Boolean

Required: Yes

Default: `false`

Alert Builder Fields and Requirements

Alert Builder Requirements - Event Fields

The minimum requirement Event Fields for the Alert Builder are listed in the table. For the full list of fields see [Event and Alert Field Reference](#).

	Field	Data Type	Size	Description	Example	Comment
1	signature	VARBINARY(binary)	767	This is a special attribute used to determine when Cisco Crosswork Situation Manager deduplicates events into alerts. It can be any combination of one or more of the attributes listed below.	host1::nagios::cpu	

To be constructed as a subset of events from a source, also see existing guidance. Constructed fields should be separated by “::” avoiding any 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).

3	source_id	TEXT(utf8)	655 35	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	192.168.1.1 07
---	------------------	------------	-----------	---	-------------------

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.

4	external_id	TEXT(utf8)	65535	Any unique identifier provided in the source event (event ID, incident ID etc.)	12345
---	--------------------	------------	-------	---	-------

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.

5	manager	TEXT(utf8)	65535	A general identifier of the event generator or intermediary (NAGIOS, SCOM, etc.)	Nagios
---	----------------	------------	-------	--	--------

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 Cisco Crosswork Situation Manager.

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 LAM that received the alerts in multi-LAM deployments.

6	source	TEXT(utf8)	655 35	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	TEXT(utf8)	655 35	Class and Type are generic classifications for the event in a hierarchy that allow	cpu

you to maintain a simple event ontologies; class then type. (Disk space: free space, Memory: max used...total available, etc.)

8	agent	TEXT(utf8)	65535	<p>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 from the CI e.g. "nagios-agent-london-7"</p> <p>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></p> <pre>{ name: "agent", rule: "\$LamInstanceName" },</pre>	Linux
9	agent_location	TEXT(utf8)	65535	<p>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</p>	New York, NY

Centre, Stuttgart Main Station, (51.407139, -0.307321) etc.)

1 agent_time
0 e

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 Cisco Crosswork Situation Manager 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

* If this field is not included you need to include 17, last_event_time

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);

1	severity	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.	5	0 clear - 5 critical
2				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		
1	description	TEXT(utf8)	65535	The main text payload of the event.	CPU Threshold exceeded:	
5						



				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.	99%
1	last_event	BIGINT(binary)	20	Internally generated	* If this field is not included you need to include 10, agent_time
7	_time			DO NOT CHANGE.	
2	custom_info	TEXT(utf8)	65535	Custom_info is a special field that is the mechanism for extending the Cisco Crosswork Situation Manager 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.

You can store simple or arbitrarily complex hierarchical JSON attributes in this field. They are basically serialized for use in the standard JSON.parse/stringify manner and the Cisco Crosswork Situation Manager UI is written to display JSON hierarchies of any complexity in a tree-view format.

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 Cisco Crosswork Situation Manager deduplicates events into Alerts. It can be any combination of one or more of the attributes listed below	host1::networks::cpu	To be constructed as a subset of events from a source, also see existing guidance

Constructed fields should be separated by “:.” avoiding any 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)	65 53 5	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,	192.168.1 .107

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

4	external_id	No	TEXT(utf8)	65 53 5	Any unique identifier provided in the source event (event ID, Incident ID etc.)	12345	Returns Null if blank
---	--------------------	----	------------	---------------	---	-------	-----------------------

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		
5	manager	No	TEXT(utf8)	65 53 5	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 Cisco Crosswork Situation Manager. 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 Cisco Crosswork Situation Manager LAM that received the alerts in multi-LAM deployments	Nagios	Returns Null if blank
6	source	Yes	TEXT(utf8)	65 53 5	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	host1	

(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.)

7	class	Yes	TEXT(utf8)	65 53 5	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)	65 53 5	The specific agent that created the event, (SCOM REST, NAGIOS SOCKET,	Linux

SNMP TRAP
NATIVE, etc.). This
is typically the
name of the agent
that facilitates the
event 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,
this is the default

```
{ name:
  "agent", rule:
  "$LamInstanceName" },
```

9	agent_location	Yes	TEXT(utf8)	65 53 5	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
---	-----------------------	-----	------------	---------------	--	--------------

1	agent_time	Yes			This is the	
---	-------------------	-----	--	--	-------------	--



0 e

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 Cisco

Crosswork
Situation Manager
server time - add
the origin time to
the custom_info for
the event. This can
be operationally
useful. e.g.

custom_info.origina

lEventTime :

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);

1	type	Yes	TEXT(utf8)	65	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	
1				53			
				5			
1	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	5	0 clear - 5 critical
2							
					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

1 3	significanc e	No	INT(binary)	11	This value is calculated by Cisco Crosswork Situation Manager Events Analyser.
--------	--------------------------	----	-----------------	----	--

Internally
generated
DO NOT CHANGE

1 4	count	No	INT(binary)	11	The reference count of deduplicated Events for each Alert.
--------	--------------	----	-----------------	----	--

Internally
generated
DO NOT CHANGE

1 5	descriptio n	Yes	TEXT(utf8)	65 53 5	The main text payload of the event. Add as much textual detail as possible. Remember a human operator will look at the	CPU Threshold exceeded: 99%
--------	-------------------------	-----	------------	---------------	--	--------------------------------------



detail and the entropy calculation works best with detailed narratives.

1 first_event No BIGINT(binary) 20
6 _time

If you set agent_time in the LAM/LAMbot to the actual epoch seconds timestamp of each event, Cisco Crosswork Situation Manager will automatically keep track of the first and last occurrence of multiple instances of the same event.

Internally generated
DO NOT CHANGE

1 last_event No BIGINT(binary) 20
7 _time

1 int_last_event No BIGINT(binary) 20
8 _time

Internally generated
DO NOT CHANGE

1411134 From agent_time
582

1 last_state_change No BIGINT(binary) 20
9

Internally generated
DO NOT CHANGE

2 state No INT(binary) 11
0

- 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

2 1	owner	No	INT(binary)	11	Set when an operator right- clicks on an alert in the Cisco Crosswork Situation Manager UI and assigns ownership.
--------	--------------	----	-----------------	----	--

Internally
generated
DO NOT CHANGE

2 2	entropy	No	DOUBLE(bi nary)	22	Internally generated DO NOT CHANGE
--------	----------------	----	--------------------	----	---

2 3	custom_in fo	No	TEXT(utf8)	65 53 5	Custom_info is a special field that is the mechanism for extending the Cisco Crosswork Situation Manager 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	Returns Null if blank
--------	-------------------------	----	------------	---------------	--	-----------------------------

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.

You can store simple or arbitrarily complex hierarchical JSON attributes in this field. They are basically serialized for use in the standard JSON.parse/stringify manner and Cisco Crosswork Situation Manager UI is written to display JSON hierarchies of any complexity in a tree-view format

Alert and Event Field Reference

This is a reference guide for alert and event fields, input types, field descriptions and output examples.

Field	Type	Description	Example Output
-------	------	-------------	----------------



active_situations	Array	Situation IDs of any Situations the alert is associated with.	1, 6, 8
agent_host	Text	Host machine or physical location of the agent that created the event.	OEM Monitor 1
agent_name	Text	Name of the agent that created the event.	NAGIOS SOCKET
agent_location	Text	Host machine or physical location of the agent that created the event.	London Data Centre (51.4167,-0.2833)
agent_time	Integer	Timestamp of when the event occurred in epoch time. Use \$moog_now in the mapping to set agent time to the time the event arrived at Cisco Crosswork Situation Manager.	1516183437
alert_id	Integer	Internal identifier generated by Cisco Crosswork Situation Manager.	101
class	Text	Level of classification for an event. This follows the hierarchy; class then type.	CISCO-IF-Extension-MIB



count	Integer	Number of events in the alert.	2
custom_info	Text	Custom information added as a JSON encoded string.	custom_info.myNodeList=["node1" , "node2" , "node3"]
description	Text	Text description of the alert.	Network Interface (ifIndex = 512479388) Up (ifEntry.52683483)
entropy	Integer	Measure of uncertainty of an outcome between 0 and 1 (0 meaning very certain and 1 meaning very uncertain).	0.4
external_id	Integer	Unique identifier from the event source.	7622183
first_event_time	Integer	Earliest event time for the alert. This is calculated from the agent_time of the events that constitute the alert.	14:08:14 16/01/2018
host	Text	Name of the source machine that generated the event.	OEM Server 2
internal_last_event_time	Integer	Time that the latest event for the Alert was received by the Moog server.	10:24:03 19/01/2018
last_change	Integer	Time that the alert was last updated in the	12:38:06 19/01/2018



Cisco Crosswork Situation Manager UI.			
last_event_time	Integer	Latest event time for the alert. This is calculated from the agent_timeof the events that constitute the alert.	10:24:03 19/01/2018
manager	Text	General identifier of the event generator or intermediary.	NAGIOS, SCOM.
owned_by	Text	Alert owner's username.	John Smith
severity	Integer	Severity level of the alert between 0 and 5.	4
significance	Integer	Relative significance of an alert is calculated based on its entropy. See Significance .	3
situations	Array	Any situations the alert is associated with, including those that have been resolved or closed.	24, 01
source	Text	Name of the source machine that generated the event. If there is no source machine or application, the source is the name of the	A hostname or fully qualified domain name (FQDN).



		instance (database name, cluster node, container name).	
source_id	Text	Identifier for the source machine that generated the event.	5dc68d65-532c-4918-be12- 21e1cbcf7af2
status	Text	Status of the alert.	Assigned
type	Text	Level of classification for an event. This follows the hierarchy; class then type.	CISCO-IF-Extension-MIB Notification

Alert Builder Outputs

These fields depend on how you configure the Alert Builder Inputs. Note that the numbers refer to the absolute list described in [Alert and Event Field Reference](#).

	Field	Data Type	Size	Description	Example	Comment
1	signature	VARBINARY (binary)	767	This is a special attribute used to determine when Cisco Crosswork Situation Manager 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 possible issues with concatenation providing misleading	host1::nagi os::cpu	

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	BIGINT(binary)	20	An auto-assigned incremental number. Internally generated DO NOT CHANGE	
3	source_id	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	192.168.1.107

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

4	external_id	TEXT(utf8)	655 35	Any unique identifier provided in the source event (event ID, Incident ID etc.)	12345
---	--------------------	------------	-----------	---	-------

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

5	manager	TEXT(utf8)	655 35	A general identifier of the event generator or intermediary (NAGIOS, SCOM, etc.)	Nagios
---	----------------	------------	-----------	--	--------

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 Cisco

Crosswork Situation Manager.

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 LAM that received the alerts in multi-LAM deployments

6	source	TEXT(utf8)	655 35	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	host1
---	---------------	------------	-----------	--	-------



				(container UUID, cluster node UUID etc.)	
7	class	TEXT(utf8)	655 35	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	TEXT(utf8)	655 35	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 from the CI e.g. "nagios-agent-london-7"	Linux
				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	TEXT(utf8)	655 35	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	New York, NY

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.)

1 **agent_time**
0

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);

1	type	TEXT(utf8)	655	Class and Type are	DOWN	
1			35	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.)		
1	severity	INT(binary)	11	Standard 0-5 but be	5	0 clear - 5
2				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		critical

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

1 3	significance	INT(binary)	11	This value is calculated by Events Analyzer.
--------	---------------------	-------------	----	--

Internally generated
DO NOT CHANGE

1 4	count	INT(binary)	11	The reference count of deduplicated Events for each Alert.
--------	--------------	-------------	----	--

Internally generated
DO NOT CHANGE

1 5	description	TEXT(utf8)	655 35	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.	CPU Threshold exceeded: 99%
--------	--------------------	------------	-----------	---	--------------------------------------

1 6	first_event_ time	BIGINT(bina ry)	20	If you set agent_time in the LAM/LAMbot to the actual epoch seconds timestamp of
--------	------------------------------	--------------------	----	---



each event, Cisco Crosswork Situation Manager will automatically keep track of the first and last occurrence of multiple instances of the same event.

Internally generated
DO NOT CHANGE

1 last_event_t BIGINT(binary) 20
7 ime

1 int_last_event_time BIGINT(binary) 20
8 nt_time

1 last_state_change BIGINT(binary) 20
9 hange

2 state INT(binary) 11
0

Internally generated 14111345 Fromagent
DO NOT CHANGE 82 t_time

Internally generated
DO NOT CHANGE

- 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

2 owner INT(binary) 11
1

Set when an operator right-clicks on an alert in the Cisco Crosswork Situation Manager UI and assigns ownership.

Internally generated



DO NOT CHANGE

2	entropy	DOUBLE(bin	22	Internally generated
2		ary)		DO NOT CHANGE
2	custom_inf	TEXT(utf8)	655	Custom_info is a
3	o		35	special field that is
				the mechanism for
				extending the Cisco
				Crosswork Situation
				Manager 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.
				 You can store simple
				or arbitrarily
				complex hierarchical
				JSON attributes in
				this field. They are
				basically serialized



for use in the standard JSON.parse/stringify manner and Cisco Crosswork Situation Manager UI is written to display JSON hierarchies of any complexity in a tree-view format

Maintenance Window Manager

- [Configure Maintenance Window Manager](#)
- [Example Configuration](#)
- [Maintenance Windows](#)
- [Updating Captured Alerts](#)

The Maintenance Window Manager Moolet compares alerts against active maintenance windows. If the alerts match an active Maintenance Schedule filter, then they are not forwarded onto the next part of the chain. This prevents a Sigaliser Moolet clustering these alerts into Situations.

Configure Maintenance Window Manager

Edit the configuration file at
\$MOOGSOFT_HOME/config/moolets/maintenance_window_manager.conf.

Refer to [Maintenance Window Manager Reference](#) to see all available properties.

Example Configuration

The following example demonstrates a simple Maintenance Window Manager configuration:

```
{
  name           : "MaintenanceWindowManager",
  classname      : "CMaintenance",
  run_on_startup : true,
  metric_path_moolet : true,
  process_output_of : "AlertBuilder",
  maintenance_status_field : "maintenance_status",
  maintenance_status_label : "In maintenance",
  update_captured_alerts : true
}
```

Maintenance Windows



You can use the Maintenance Schedule functionality to schedule outages when you do not want new Situations to be created from these alerts. You can configure the Maintenance Manager Moolet to ensure that alerts are not passed along to Sigalisers and clustered into Situations during that time period. You can set up maintenance windows using:

- UI: See [Schedule Maintenance Downtime](#) for more information on how to set up maintenance windows. [Schedule Maintenance Downtime](#)
- [Graze API](#). Graze API

Updating Captured Alerts

In addition to implementing the maintenance windows, the Maintenance Window Manager Moolet updates the following `custom_info` fields in each alert affected by a maintenance window. Because the Maintenance Window Manager uses these `custom_info` fields within the alerts, Moobots must not overwrite these `custom_info` fields or completely empty the `custom_info` object within alerts.

Field	Description
<code>custom_info.maintenance_status</code>	Configurable text label. Set to "In maintenance" by default.
<code>custom_info.maintenance_id</code>	Numerical ID of the maintenance window that captured the alert.
<code>custom_info.maintenance_name</code>	Name of the maintenance window that captured the alert.
<code>custom_info.forward_Alerts</code>	Whether the alert is forwarded to Sigalisers or not. Set to false by default.

Maintenance Window Manager Reference

This is a reference for the [Maintenance Window Manager](#) Moolet.

Cisco recommends you do not change any properties that are not in this reference guide.

You can change the behavior of the Maintenance Window Manager by editing the configuration properties in the `$MOOGSOFT_HOME/config/moolets/maintenance_window_manager.conf` configuration file. It contains the following properties:

name

Name of the Maintenance Window Manager Moolet. Do not change.

Type: String

Required: Yes



Default: "MaintenanceWindowManager"

classname

Moolet class name. Do not change.

Type: String

Required: Yes

Default: "CMaintenance"

run_on_startup

Determines whether the Maintenance Window Manager runs when Cisco Crosswork Situation Manager starts. By default, it is set to true, so that when Moogfarmd starts, it automatically creates an instance of the Maintenance Window Manager.

Type: Boolean

Required: Yes

Default: true

metric_path_moolet

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

Type: Boolean

Required: Yes

Default: true

process_output_of

Defines the input source for the Maintenance Window Manager. This determines the Maintenance Window Manager's place in the alert processing workflow.

Type: List

Required: Yes

One of: AlertBuilder, AlertRulesEngine, Enricher

Default: "AlertBuilder"

maintenance_status_field

Name of the custom_info field or key used to indicate the alert's maintenance status.

Type: String

Required: Yes



Default: "maintenance_status"

maintenance_status_label

Value of the `custom_info.maintenance_status` field used to indicate that an alert is in maintenance.

Type: String

Required: Yes

Default: "In maintenance"

update_captured_alerts

If enabled, ensures the maintenance status of an alert is set to null once the Maintenance Window that captured it has expired. If disabled, the maintenance status field of a captured alert remains as the text value set in the `maintenance_status_label` property, unless that alert reoccurs when all `custom_info` maintenance fields are set to null.

Type: Boolean

Required: Yes

Default: true

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 `maintenance_status_label` in the MaintenanceWindowManager Moolet configuration in `$MOOGSOFT_HOME/config/moolets/maintenance_window_manager.conf`.

For the feature to function, you must place the Maintenance Window Manager Moolet before a Sigalising Moolet in a forwarding chain. It is also appropriate for you to locate it before the Alert Rules Engine in the processing chain.

Alert Rules Engine

The Alert Rules Engine uses business logic to process alerts based on certain conditions. The conditions that the Alert Rules Engine works with generally involve a time-based analysis so that it can process an event in the context of events that happen later. You can define rules in the Alert Rules Engine to hold alerts for a period of time, identify missing alerts or change the state of alerts. For example, common uses of the Alert Rules Engine include:

- **Link Up-Link Down:** Delays an alert to see if a link recovers.
- **Heartbeat Monitor:** Detects any missing network health signals.
- **Closing Events:** Closes events of a particular type or severity.
- **Merging:** Merges the state of two distinct alerts.

Configure Alert Rules Engine



Edit the configuration file at

\$MOOGSOFT_HOME/config/moolets/alert_rules_engine.conf.

Refer to [Alert Rules Engine Reference](#) to see all available properties.

Example Configuration

The following example demonstrates a simple Alert Rules Engine configuration:

```
{
  name           : "AlertRulesEngine",
  classname      : "CAAlertRulesEngine",
  run_on_startup : false,
  metric_path_moolet : true,
  moobot        : "AlertRulesEngine.js",
  process_output_of : "MaintenanceWindowManager"
}
```

Define Action States and Transitions

The Alert Rules Engine uses Action States and transitions and their properties, to process alerts through business logic defined in the `AlertRulesEngine.js` Moobot. 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.

See [Action States](#) and [Transitions](#) for further information on how to define them and the properties available.

The initial state for all alerts is the 'Ground' state. After an alert enters 'Ground' state, the Alert Rules Engine 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, the Alert Rules Engine 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, the Alert Rules Engine 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, the Alert Rules Engine does not return it to 'Ground' state and it is closed.

Note

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

Alert Rules Engine Examples

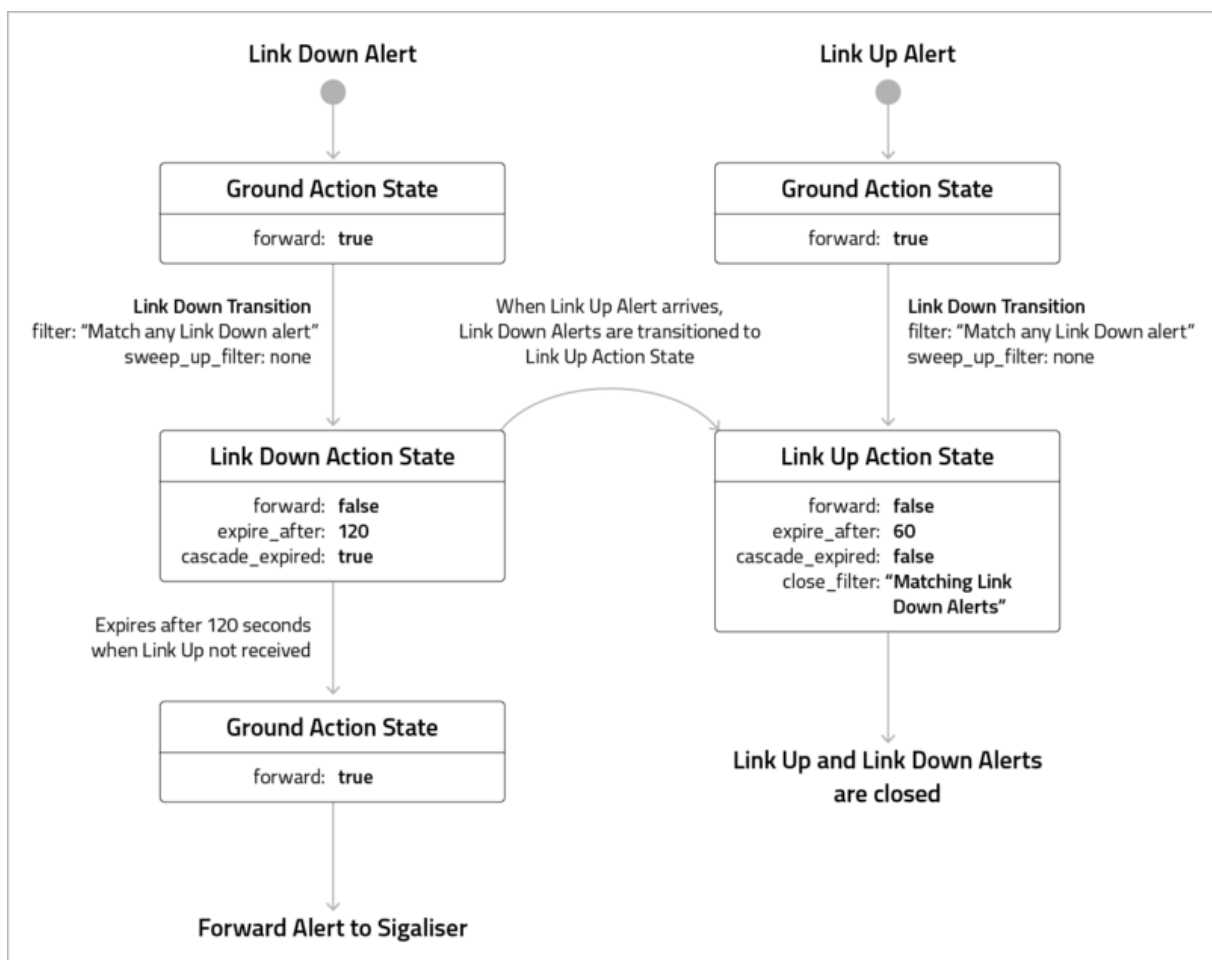


The Alert Rules Engine can be set up to process [Link Up-Link Down](#) events. It can also be set up to act as a [Heartbeat Monitor](#).

Link Up-Link Down Example

This example demonstrates how to configure the Alert Rules Engine so that when a Link Down alert arrives at Moogfarmd, the Alert Rules Engine holds it for a period of time to provide an opportunity for the Link Up alert to arrive. If nothing arrives, the Alert Rules Engine forwards it to a Sigaliser.

If the Link Up alert arrives, the Alert Rules Engine closes and discards both alerts without sending anything to the Sigaliser. This ensures that neither the Link Down nor the Link Up alert 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: 'Link Down Transition' and 'Link Up Transition'.

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



Heartbeat Monitor

You can configure the Alert Rules Engine Moolet 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, Cisco Crosswork Situation Manager creates a Situation when an event source does not send a heartbeat after a given time period. The Alert Rules Engine 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 (clustering algorithm).

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 the [Alert Rules Engine](#) Moolet, [Action States](#) and [Transitions](#) for further details.
- You can identify heartbeat alerts in the integration by description, class or another configurable field. These must be specific, regular events that arrive at 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, 'type' is 'heartbeat' in the Alert Rules Engine trigger filter and 'class' is 'heartbeat' in the Cookbook Recipe trigger filter.

Create a Heartbeat Monitor

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

1. Edit `$MOOGSOFT_HOME/bots/moobots/AlertRulesEngine.js` and add the `heartBeatSeverity` exit action. This function changes the alert severity to critical and ensures alerts that are closed are not forwarded to the Cookbook. See [Status ID Reference](#) for a list of status IDs.

```
function heartBeatSeverity(alert,associated) {
  var currentAlert = moogdb.getAlert(alert.value("alert_id"));
  if ( currentAlert && currentAlert.value("state") !== 9 ) {
    alert.set("severity",5);
    var alertDescr = currentAlert.value("description");
    // Update the description to "MISSED", a successful heartbeat will
    reset this.
    if ( !/^MISSED/i.test(alertDescr) ) {
      alert.set("description", "MISSED: " + alertDescr)
    }
  }
}
```



```
        moogdb.updateAlert(alert);
        currentAlert.forward("HeartbeatCookBook");
    }
}
```

2. Navigate to **Settings > Action States** in the Cisco Crosswork Situation Manager UI. 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

Warning

* The Remember alerts for setting is the timer. Set this to two or three times your heartbeat interval time.

3. Go to **Settings > Transitions** in the Cisco Crosswork Situation Manager UI. 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 (manager = "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':



4. Ensure Alert Rules Engine is enabled. To do this, edit the `$MOOGSOFT_HOME/config/moolets/alert_rules_engine.conf` file and set `run_on_startup` to true.
5. Create a `heartbeat.conf` configuration file in `$MOOGSOFT_HOME/config/moolets` to add a Heartbeat Cookbook for heartbeat alerts. This only works with these alerts:

```
# Moolet
  name:"HeartbeatCookBook",
  classname:"CCookbook",
  run_on_startup:true,
  metric_path_moolet : true,
  moobot:"Cookbook.js",
  process_output_of:"[]",
  # 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:"class = 'heartbeat' AND agent = 'SCOM'",
        rate:0,
        # Given in events per minute
        min_sample_size:5,
        max_sample_size:10,
        matcher:{
            components:[
                {
                    name:"agent",
                    similarity:1.0
                }
            ]
        }
    },
    {
        chef:"CValueRecipe",
        name:"ScomHeartbeatChange",
        description:"SCOM Heartbeat: Cluster host change",
        recipe_alert_threshold:0,
        exclusion:"state = 9",
        trigger:"class = 'heartbeatRoleChange' AND agent = 'SCOM'",
        rate:0,
        # Given in events per minute
        min_sample_size:5,
        max_sample_size:10,
        matcher:{
            components:[
                {
                    name:"agent",
                    similarity:1.0
                }
            ]
        }
    }
],
cook_for:20000
}
```

Save heartbeat.conf.

6. Edit the Moogfarmd configuration file \$MOOGSOFT_HOME/config/moog_farmd.conf to add a new merge group that references the HeartBeatCookbook Moolet. Configure this merge group to have an alert_threshold of 1 to allow a single alert to create a Situation (by default, a minimum of 2 alerts are required to create a Situation):

```
merge_groups:
[
    {
        name: "Heartbeat",
        moolets: ["HeartbeatCookBook"],
```



```
        alert_threshold : 1,  
        sig_similarity_limit : 1  
    }  
],
```

7. Include the Moolet configuration by adding the following in `$MOOGSOFT_HOME/config/moog_farmd.conf`:

```
{  
    include : "heartbeat.conf"  
},
```

Save the changes to `moog_farmd.conf`.

8. 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 status of alerts and Situations is determined by their status ID. These statuses are used within the [Heartbeat Monitor](#).

The different `status_id` values are as follows:

Status ID	Name
-----------	------



1	Opened
2	Unassigned
3	Assigned
4	Acknowledged
5	Unacknowledged
6	Active
7	Dormant
8	Resolved
9	Closed
10	SLA Exceeded

Alert Rules Engine Reference

This is a reference for the [Alert Rules Engine](#) Moolet.

Cisco recommends you do not change any properties that are not in this reference guide.

You can change the behavior of the Alert Rules Engine by editing the configuration properties in the `$MOOGSOFT_HOME/config/moolets/alert_rules_engine.conf` configuration file. It contains the following properties:

name

Name of the Alert Rules Engine Moolet. Do not change.

Type: String

Required: Yes

Default: "AlertRulesEngine"

classname

Moolet class name. Do not change.

Type: String

Required: Yes

Default: "CAAlertRulesEngine"

run_on_startup

Determines whether the Alert Rules Engine runs when Cisco Crosswork Situation Manager starts. By default, it is set to `false`, so it does not start when Moogfarmd starts.



You can change this property to `true` so that, when Moogfarmd starts, it automatically creates an instance of the Alert Rules Engine.

Type: Boolean

Required: Yes

Default: `false`

metric_path_mooret

Determines whether or not Cisco Crosswork Situation Manager includes the Alert Rules Engine in the Event Processing metric for [Self Monitoring](#). Self Monitoring

Type: Boolean

Required: Yes

Default: `true`

moobot

Specifies a JavaScript file found in `$MOOGSOFT_HOME/moobots`, which defines the Alert Rules Engine Moobot. The default, `AlertRulesEngine.js`, provides the standard modules. You can customize it to meet your needs.

Type: String

Required: Yes

Default: `"AlertRulesEngine.js"`

mooms_event_handler

Determines whether or not the Alert Rules Engine listens for messages on the message bus. If set to `true`, the Alert Rules Engine processes messages on the Alerts topic on the message bus. This property should not be included in the configuration file, or should be commented out, if the `process_output_of` property is defined.

Type: Boolean

Required: No

Default: `false`

process_output_of

Defines the input source for the Alert Rules Engine. This determines the Alert Rules Engine's place in the alert processing workflow. If this property is defined, the `mooms_event_handler` property should be omitted or commented out in the configuration file.

Type: List

Required: No



One of: AlertBuilder, MaintenanceWindowManager, Enricher

Default: "MaintenanceWindowManager"

Empty Moolet

The Empty Moolet enables Cisco Crosswork Situation Manager integrators to intercept and handle Message Bus events without impacting upon the existing alert flow logic and processing. This provides a mechanism for you to implement your own alert processing rules. The Empty Moolet can also be used to provide general augmentation of alert and Situation details, for example, [enrichment](#).

An Empty Moolet can be passed an alert or a Situation by one of the following mechanisms:

- Process output of: The Empty Moolet exists in the alert processing chain.
- Event handler: The Empty Moolet listens for specific message types on the bus.
- Direct forwarding: The Empty Moolet is handed an object by another Moolet, for example, Moolet A forwards an alert to Moolet B.

A single Empty Moolet uses one or more of these mechanisms.

Configure Empty Moolet

The Empty Moolet takes messages off the Message Bus according to message type and passes them to a Moolet. The configuration includes the message types to register interest for and the name of the Moolet to pass them to. For example, to integrate with an incident management system such as ACMEIncidentManager, the Empty Moolet must:

- Listen to NewThreadEntry events (the topic on Message Bus is /sig/thread/entry/new) and SigStatus events (the topic on Message Bus is /sigs/status topic).
- 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.REST.V2

The following example demonstrates an Empty Moolet configuration for this scenario:



```
{
    name          : "ACMEIncidentManager",
    classname      : "CEmptyMoolet",
    run_on_startup : true,
    moobot         : "ACMEIntegration.js",
    event_handlers : [
        "NewThreadEntry".
        "SigStatus"
    ]
}
```

This example shows one way of integrating Cisco Crosswork Situation Manager with another system. Each integration is dependent upon the individual use cases and systems being integrated.

See [Alert Manager](#) for a further example of an Empty Moolet configuration.

Note

Not all event handlers are required for every integration. Only specify required handlers.

Customize Empty Moolet

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.REST.V2

Edit the configuration file `moog_farmd.conf` to associate the CustomTaskRunner Moobot with the Empty Moolet, and listen for SigAction events:

```
{
    name          : "CustomTaskRunner",
    classname      : "CEmptyMoolet",
    run_on_startup : true,
    metric_path_moolet : false,
    moobot         : "CustomTaskRunner.js",
    event_handlers : [
        "SigAction"
    ]
}
```

This is an 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.



Alert Manager

- [Configure Alert Manager](#)
- [Example Configuration](#)
- [Alert Manager Moobot](#)
- [Empty Moolet](#)

The Alert Manager uses the Empty Moolet to enable Cisco Crosswork Situation Manager administrators or implementers to incorporate additional alert processing not handled by the Alert Builder, Maintenance Window Manager or Alert Rules Engine. You can use the Alert Manager in standalone mode or as part of the alert processing workflow.

Configure Alert Manager

Edit the configuration file at `$MOOGSOFT_HOME/config/moolets/alert_manager.conf`.

See [Empty Moolet Reference](#) for a full description of all properties. Some properties in the file are commented out by default. [Empty Moolet Reference](#)

You can use the following mechanisms to determine Alert Manager behavior:

- If `standalone_moolet = true`: The Alert Manager picks up alerts, specified by the `event_handlers` property, on the Message Bus and processes them.
- If you set `process_output_of` to Maintenance Window Manager or Alert Rules Engine: The Alert Manager uses the output of that component.

Example Configuration

The default configuration file contains an example implementation of the Empty Moolet functionality in the form of the Alert Manager Moolet. For example:

```
{
  name           : "AlertMgr",
  classname      : "CEmptyMoolet",
  run_on_startup : false,
  metric_path_moolet: false,
  moobot         : "AlertMgr.js",
  standalone_moolet : true,
  # Listens for alerts events (on the /alerts topics)
  event_handlers : [
    "AlertClose",
    "AlertUpdate",
    "Alert"
  ]
}
```

Alert Manager Moobot

Cisco provides a Moobot for the Alert Manager Moolet named `AlertMgr.js`. An example use case for this Moolet is to enable a specific action on different alert types. For



example, to update a Situation's services when an alert is updated which contains certain attributes.

Empty Moolet

For further information on customizing Cisco Crosswork Situation Manager using the Empty Moolet, see [Empty Moolet](#).

Empty Moolet

The Empty Moolet enables Cisco Crosswork Situation Manager integrators to intercept and handle Message Bus events without impacting upon the existing alert flow logic and processing. This provides a mechanism for you to implement your own alert processing rules. The Empty Moolet can also be used to provide general augmentation of alert and Situation details, for example, [enrichment](#).

An Empty Moolet can be passed an alert or a Situation by one of the following mechanisms:

- Process output of: The Empty Moolet exists in the alert processing chain.
- Event handler: The Empty Moolet listens for specific message types on the bus.
- Direct forwarding: The Empty Moolet is handed an object by another Moolet, for example, Moolet A forwards an alert to Moolet B.

A single Empty Moolet uses one or more of these mechanisms.

Configure Empty Moolet

The Empty Moolet takes messages off the Message Bus according to message type and passes them to a Moolet. The configuration includes the message types to register interest for and the name of the Moolet to pass them to. For example, to integrate with an incident management system such as ACMEIncidentManager, the Empty Moolet must:

- Listen to NewThreadEntry events (the topic on Message Bus is `/sig/thread/entry/new`) and SigStatus events (the topic on Message Bus is `/sigs/status` topic).
- 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.REST.V2

The following example demonstrates an Empty Moolet configuration for this scenario:

```
{
    name                : "ACMEIncidentManager",
    classname           : "CEmptyMoolet",
    run_on_startup      : true,
    moobot              : "ACMEIntegration.js",
    event_handlers      : [
        "NewThreadEntry".
        "SigStatus"
    ]
}
```

This example shows one way of integrating Cisco Crosswork Situation Manager with another system. Each integration is dependent upon the individual use cases and systems being integrated.

See [Alert Manager](#) for a further example of an Empty Moolet configuration.

Note

Not all event handlers are required for every integration. Only specify required handlers.

Customize Empty Moolet

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.REST.V2

Edit the configuration file `moog_farmd.conf` to associate the CustomTaskRunner Moobot with the Empty Moolet, and listen for SigAction events:

```
{
    name                : "CustomTaskRunner",
    classname           : "CEmptyMoolet",
    run_on_startup      : true,
    metric_path_moolet  : false,
    moobot              : "CustomTaskRunner.js",
    event_handlers      : [
        "SigAction"
    ]
}
```

This is an 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.

Enrichment

Situations in Cisco Crosswork Situation Manager are built from data ingested from your monitoring systems. You may have use cases for your Situations that require more information than is contained in the raw data. If this is the case, you can use a process called enrichment to add supplemental data to alerts or Situations. Enrichment can:

- Improve accuracy for clustering alerts into Situations.
- Improve readability of alerts for operators.
- Aid operators in investigating Situations.
- Provide critical reporting data.

The first step is to identify whether your existing data is sufficient. If it is lacking, identify the type of enrichment data that meets your requirements and the data source that can provide it. You can then choose the most effective and efficient method of enrichment for your specific needs.

Do You Need To Enrich?

The need to enrich depends on whether the data from your data source or monitoring system fulfills your requirements. Examine the use cases for your data to identify any omissions.

For example, an organization sets up Cisco Crosswork Situation Manager to ingest the following event data:

```
"node_name"           : "U0039-router01"  
"description" : "Router down"
```

The data must fulfill these use cases:

1. Operators need the site name to understand where they need to take action to fix the problem.
2. Management needs the region for reporting requirements.

For this company, the node names are all based on the site name `<site>-<component>` so "U0039" reflects the site. There is no need to enrich for this use case.

The site name is not enough to determine the region, and the event data does not include region data. To satisfy the second use case, the company needs to enrich the event data.



Identify the Enrichment Purpose

The purpose of the enrichment indicates whether to enrich at alert or Situation level. Enrichment is expensive in terms of processing time and resource use. Inefficient enrichment can slow the processing of alerts, so it is important to enrich at the appropriate level.

Enrichment data can be broadly categorized to fulfil one of the following purposes:

- **Operational:** Functionally modifies behavior within Cisco Crosswork Situation Manager to drive processes such as clustering. Ideally performed on alert creation.
- **Informational:** Assists a consumer (operator or external system) to differentiate between Situations. Typically performed at Situation level. Includes updates to Situation description, services and processes.
- **Diagnostic:** Assists operators to investigate Situations and can be performed at either alert or Situation level. Examples include updates to alert and Situation custom_info and updates to Situation discussion threads.

The region data in our example is informational.

Identify the Enrichment Source

If the required data exists externally, identify its type:

- **Static:** Data that changes infrequently, for example a country code lookup to a country name.
- **Dynamic:** Data that may be subject to change, for example a database query to match a hostname to a service.

In our example, the company database stores the site number and relates it to the site address and region. The data is static:

site address city state region

U0039 1265 Battery St San Francisco CA US-WEST

Dynamic enrichment on every de-duplication has a greater performance impact than enrichment on alert creation. If the enrichment data is unlikely to change during the lifetime of an alert, enrich once on alert creation. See [Enrich on Alert Creation](#) for more details.

You can enrich from a static file in a LAMbot. All other enrichment is performed in a Moobot.

Select an Enrichment Method

Some enrichment methods are available in the UI:

- [Situation Room Plugins](#).



- [UI Enrichment](#) using a static data file. UI Enrichment
- [Link Definitions](#). Create Link Definitions

Other methods are manually configured or accessed via the command line. The most common are:

- [REST.V2](#) module to retrieve data through HTTP. REST.V2
- [ExternalDb](#) module to retrieve data from supported SQL databases. ExternalDb
- [Graze API](#) to update Situations and alerts statically. Graze API
- [Situation Manager Labeller](#) to update Situations and alerts dynamically.

In our example, depending on the database specification, the company might use JDBC to add the region data into alert custom_info and the Situation Manager Labeller to add the region data to Situations.

Enrich on Alert Creation

If your enrichment data is unlikely to change during the lifetime of an alert, enrich once on alert creation.

To enrich on alert creation:

- Create a custom alert enricher Moolet.
- Configure your alert enricher to use caching.
- Configure the Alert Builder to send data to your custom Moolet on alert creation.
- Define your custom Moolet in Moogfarmd.

See [Enrichment](#) for more information on enrichment methods and processes.

Create an Alert Enricher Moobot

Create an Alert Enricher Moobot to obtain enrichment data from your external source, for example via [JDBC](#). JDBC

Use Caching

The Bot utility is included with the Situation Manager Labeller available from Cisco Support. See [Install the Situation Manager Labeller](#) for more information.

You can configure your Alert Enricher Moobot to use the caching facilities in the Bot utility. This is optional but good practice if the data is relatively static. It reduces the time required to repeatedly process data from a third party system. For example:

```
var USE_CACHE = false;
var CMDB_CACHE_RETENTION = 3600;

if (USE_CACHE && cmdb_cache_exists && cmdb_cache_exists.enrichment)
```



```
{
    customInfo.enrichment = cmdb_cache_exists.enrichment;
}
else
{
    botUtil.addObject(customInfo, "enrichment", ci_enrichment, false);
    var cmdb_cache = {};
    cmdb_cache.enrichment = customInfo.enrichment;
    botUtil.setCacheValue(botModules.constants, "CMDB"+host,
cmdb_cache, CMDB_CACHE_RETENTION);
}
```

Configure the Alert Builder

In the following example the Alert Builder sends newly created alerts to the Alert Enricher Moolet and updated alerts to the Maintenance Window Manager:

```
if(alert)
{
    var alertAction=alert.payload().getAction() === "Alert Created" ?
"create" : "update";

    if ( alertAction === "create" ) {
        logger.info("createAlert: Created Alert Id: " +
alert.value("alert_id"));
        alert.forward("AlertEnricher");
    }
    else {
        logger.info("createAlert: Updated Alert Id: " +
alert.value("alert_id"));
        alert.forward("MaintenanceWindowManager");
    }
}
```

Configure Moogfarmd

Define the Alert Enricher Moobot in Moogfarmd. For example:

```
{
    name                : "AlertEnricher",
    classname           : "CEmptyMoolet",
    run_on_startup      : true,
    persist_state       : true,
    metric_path_moolet  : false,
    moobot              : "AlertEnricher.js",
    standalone_moolet   : true,
    threads             : 5
}
```

Enricher Moolet

Enable the Enricher Moolet to update alerts with Enrichment data. See [UI Enrichment](#) for further information.



Configure Enricher

You can define the behavior of the Enricher Moolet by editing the `$MOOGSOFT_HOME/config/moolets/enricher.conf` configuration file.

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

metric_path_moolet

Determines whether or not Enricher is included in the the Event Processing metric for [Self Monitoring](#).Self Monitoring

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"
}
```

Note

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

[Moogfarmd Reference](#)

Moogfarmd controls all other services in Cisco Crosswork Situation Manager and manages which algorithms and Moolets are running.

[Services](#)

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

`/etc/init.d/moogfarmd`

A backup Moogfarmd service script is located at `$MOOGSOFT_HOME/etc/service-wrappers/moogfarmd`.

If using multiple instances of Moogfarmd on the same host, we advise that you copy and modify the default Moogfarmd service script for each Moogfarmd running on the host.

[Run the Moogfarmd Service Daemon](#)

Moogfarmd is a command line executable that can be run as a service daemon.

To execute the daemon and view available arguments run:

`moog_farmd --help`

By default, you do not need either 'config' or 'instance'. If you run the system without configuring either of these, the moogfarmd instance loads the default configuration file for moogfarmd, and responds to farmd_ctrl with no instance specified. See [High Availability Overview](#) for more information on High Availability.

The moog_farmd command line executable accepts the following arguments:

Option	Input	Description
<code>--clear_state</code>	-	Clears any persisted state information associated with Moogfarmd on startup.
<code>--cluster <arg></code>	String: <cluster name>	Name of the High Availability (HA) cluster. Overwrites the value in the configuration file.



--config <arg>	String: <file path/name>	Name and path of the configuration file specific to the running Moogfarmd instance.
--group <arg>	String: <group name>	Name of the HA group. Overwrites the value in the configuration file.
-h, --help	-	Displays all command line options.
--instance <arg>	String: <instance name>	Enables you to name the Moogfarmd instance. You can refer to this name in the farmd_ctrl utility, which allows you to start, restart and reload the various Moolets.
--logconsole	-	Instructs Moogfarmd to write logs to the console only.
--logfilename <arg>	String: <file path/name>	Name and path of the Moogfarmd log file.
-l, --loglevel <arg>	INFO WARN ALL	Specifies the debug level. Defaults to WARN, which is the recommended level in all production implementations.
--mode <arg>	String: active/passive	Starts the process in passive or active mode. The default is active.
--service_instance <arg>	String: <service suffix>	Suffix for the service name.
-v, --version	-	Displays the Moogfarmd version number.

Configuration

You can control Moogfarmd behavior through the following files:

- system.conf: the general Cisco Crosswork Situation Manager system configuration file is located in \$MOOGSOFT_HOME/config/system.conf. See [System Configuration](#).
- moog_farmd.conf: configuration specific to Moogfarmd operation. If you run multiple instances of Moogfarmd, each needs its own configuration file. All instances of Moogfarmd which do not specify a different --config use the default configuration file located in \$MOOGSOFT_HOME/config/moog_farmd.conf.

Moogfarmd runs individual isolated applications called [Moolets](#) inside the Moogfarmd app container. Moolets are a parallel concept to servlets in a traditional enterprise application container such as Tomcat. Moogfarmd controls the flow of data through the Moolets where the data can come via the Message Bus or from other Moolets.



Moolets

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 Moolet behavior.

Events can trigger a Moolet in Moogfarmd as follows:

- `process_output_of`: The Moolet listens for events from another named Moolet. You can use this method to chain Moolets together to form an automated workflow pipeline.
- `mooms_event_handler`: The Moolet listens for events on the Message Bus, for example actions triggered by a user or within another instance of moogfarmd.
- `standalone_moolet`: The Moolet listens for events generated by other Moolets within the same Moogfarmd instance without being part of the same process chain.
- `scheduler`: A unique Moolet type that allows time based task execution.

Refer to the documentation on individual Moolets to learn about how to configure their behavior:

Classic Sigaliser Moolet

Warning

Support for Classic Sigaliser will be deprecated in the release of Cisco Crosswork Situation Manager v7.3.0. If you want to use time-based clustering see [Tempus](#).Tempus

The Sigaliser Moolet, 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 Message Bus.

Basic Concepts

You can configure and tune Sigaliser Classic by editing the parameters in the `$MOOGSOFT_HOME/config/moolets/sigaliser.conf` configuration file. 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 Cisco Crosswork Situation Manager 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

You can define the Sigaliser behavior in the Sigaliser section of the Moogfarmd configuration file. 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 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 properties. The name, classname, and run_on_startup properties are shared with other Moolets.

```
{
    name                : "Sigaliser",
    classname            : "CSigaliser",
    run_on_startup       : false,
    process_output_of    : "AlertBuilder"
}
```

name

The 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 Moogfarmd starts, it does not automatically create an instance of the Sigaliser. In this case you can start it using farmd_ctrl.

Undertaking the sigalising

These properties in the Moolet direct which output should be processed:



process_output_of

Instructs the Moolet to process the output of the Alert Builder or Alert Rules Engine. Usually the Sigaliser connects directly to the Alert Builder, and the Alert Rules Engine is only used if automations are desired prior to Situation resolution. The Sigaliser can have only one input.

Algorithmics

The Sigaliser runs the matrix factorization algorithms, the properties for which are as follows:

```
# 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. We recommend that you set `time_compression` to true for low data rates and false for normal data rates.

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. We recommend 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.



The `membership_limit` property restricts the number of Situations in which an alert can appear. As Situations become 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:

$$JSC = 2 \text{ (common alerts)} / [1 \text{ (unique to Situation 1)} + 2 \text{ (common to both)} + 1 \text{ (unique to Situation 2)}] = 2 / (1+2+1) = 2/4 = \mathbf{0.5}.$$

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. We recommend 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 Cisco Crosswork Situation Manager 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.

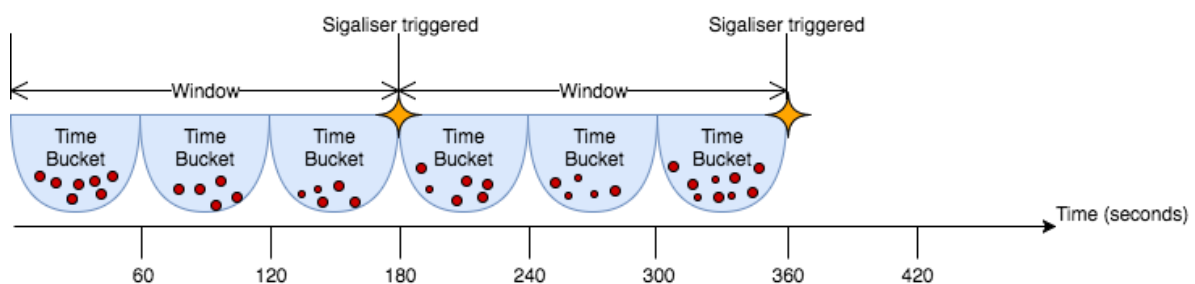
window

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.

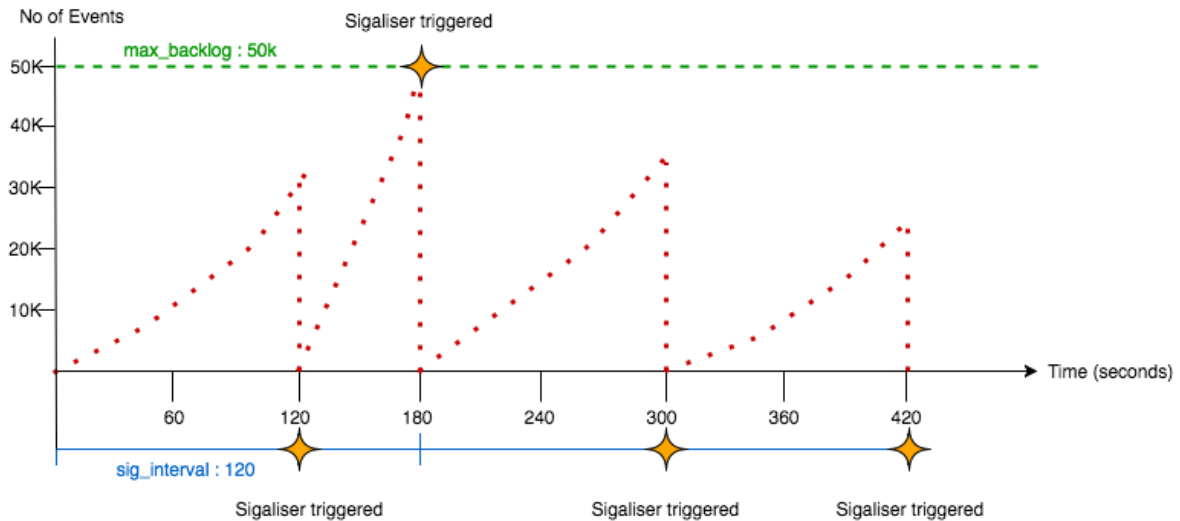
In general, for a high data rate you would use a smaller resolution and window than for a low data rate. For a fixed data rate, a smaller resolution will generally result in more Situations.

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:



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 [Configure Historic Data Retention](#) 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_cntl -v
```

Configure Housekeeper

You can define the behavior of the Housekeeper Moolet by editing the `$MOOGSOFT_HOME/config/moolets/housekeeper.conf` configuration file.

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

metric_path_moolet

Determines whether Housekeeper is factored into the event processing metric for [Self Monitoring](#) or not. See [Moogfarmd Reference](#) for more information.

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.housekeeper.CHousekeeper",
    run_on_startup      : true,
    metric_path_moolet  : false,
    standalone_moolet   : true
}
```

Note

name and classname are hardcoded and should not be changed.

Notifier Moolet

Introduction

The Notifier Moolet enables Cisco Crosswork Situation Manager to act on invite MooMS 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.



You can edit the Notifier in the `$MOOGSOFT_HOME/config/moolets/notifier.conf` configuration file. The default configuration is as follows:

```
{
  name           : "Notifier",
  classname      : "CNotifier",
  run_on_startup : false,
  metric_path_moolet : false,
  moobot         : "Notifier.js"
}
```

Scheduler Moolet

- [Scheduling Frequency](#)
- [Constraints](#)

You can schedule jobs at regular intervals by editing the `$MOOGSOFT_HOME/config/moolets/scheduler.conf` configuration file:

```
# 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,
  metric_path_moolet : false,
  moobot         : "Scheduling.js",
  threads        : 1
}
```

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 Moogfarmd 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 `$MOOGSOFT_HOME/config/moolets/situation_manager.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 starts or not.

Type: Boolean

Default: false

metric_path_moolet

Determines whether or not Situation Manager is included in the Event Processing metric for [Self Monitoring](#). Self Monitoring

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 Moolet 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"
    ]
}
```

Note

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. Constants

You can listen for specific Situation actions using the SigAction event. It can filter out the following actions:

Assigned Moderator	Deacknowledged Situation	Described Situation
Situation Resolved	Moderator	Excluded User
Situation Revived	Added Alerts To Situation	Invited User
Situation Closed	Added Entry To Thread	Moved Alerts To
Assigned Queue	Changed Situation	Situation



Created By Merge	Processes	Removed Alerts From
Used In Merge	Changed Situation	Situation
Created By Split	Services	Situation Teams Changes
Used For Split	Created Thread	Marked Thread Entry As
Ran Tool	Agreed With Thread Entry	Resolving
Acknowledged Situation	Commented On Thread Entry	Unmarked Thread Entry
Moderator	Disagreed With Thread	As Resolving
	Entry	Situation Rating
	Changed Situation Custom	Situation Rating
	Info	Removed

There are three Moobots available in `$MOOGSOFT_HOME/bots/moobots` that Situations can be sent to by Situation Manager. You can customize these 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](#).Moobot Modules

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 [Netcool Legacy LAM](#).Netcool Legacy LAM

Template Matcher Moolet

Warning

Template Matcher was deprecated from the release of Cisco Crosswork Situation Manager V7.0.0. See [Deprecation Notice: Template Matcher](#) for more information.

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.

Moogfarmd

The TemplateMatcher Moolet should be activated in Moogfarmd if you wish your system to discover "Priority Templates". By default, Moogfarmd always suppresses the creation of SPAM templates, regardless of the Moolet they are generated by.

There is no special configuration required for the Moolet; it is either "on" or "off". TemplateMatcher should run after the Alert Builder or Alert Rules Engine depending



upon the system configuration. The default configuration for the TemplateMatcher is shown below:

```
{
  name           : "TemplateMatcher",
  classname      : "CTemplateMatcher",
  run_on_startup : false,
  #process_output_of : "AlertRulesEngine",
  process_output_of : "AlertBuilder",
  threads        : 4
}
```

Standalone Execution

The TemplateMatcher Moolet can be executed in a “stand-alone” mode via the moog_primer/sigaliser utility using the following command:

```
sigaliser --moolet=TemplateMatcher
```

Teams Manager Moolet

The Teams Manager Moolet 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 Moolet can no longer act on it even if that Situation matches a filter. Graze API MoogDb V2

You can alter the behavior of the Teams Manager Moolet by changing the "Situation Update Policy" in the UI under **Settings > Teams**.

One Teams Manager Moolet is run for every instance of Cisco Crosswork Situation Manager.

Configure Teams Manager

You can configure the Teams Manager Moolet in the \$MOOGSOFT_HOME/config/moolets/teams_manager.conf configuration file.

Teams Manager Properties

The properties that relate to the Teams Manager Moolet are:

run_on_startup

Determines whether Teams Manager runs when Cisco Crosswork Situation Manager starts. If you enable it, Teams Manager processes Moolet output from the moment the system starts, without you having to configure or start it manually.



Type: Boolean

Default: true

metric_path_moolet

Determines whether or not Cisco Crosswork Situation Manager includes Teams Manager in the Event Processing metric for [Self Monitoring](#). Self Monitoring

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,
metric_path_moolet  : false,
moobot              : "TeamsMgr.js",
#
# 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   : [
    "Speedbird",
    "Cookbook",
    "Default Cookbook",
    "SituationMgr"
]
```

Note

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, Speedbird, Cookbook, Default Cookbook, SituationMgr

Default: ["Sigaliser", "Speedbird", "Cookbook", "Default Cookbook", "SituationMgr"]

Services

In Cisco Crosswork Situation Manager a service represents a supportable unit that provides a set of related functionality. A service may relate to a single application or it may incorporate multiple applications. Example services may include web application, web service, data management, database, network.

This document outlines how to create services, assign them to Situations, associate them with teams and monitor affected services in the Cisco Crosswork Situation Manager UI.

Before You Begin

Before you begin to create services in Cisco Crosswork Situation Manager, ensure you have met the following requirements:

1. Identify the services in your environment. A third party tool or external system may be useful for this task, for example the list of business services, applications or assignment groups in ServiceNow.
2. If your service data is held externally to Cisco Crosswork Situation Manager, identify the data source.
3. Choose one or more methods that you will use to create and assign services:
 - **Graze API:** Useful when you have a known list of services that change infrequently.
 - **Situation Manager Labeller:** Useful when your services are likely to change and you want to avoid the overhead associated with manual creation and assignment.
 - **Moobot:** Useful when you are already using a custom Moobot for enrichment. See [Enrichment](#) for more information.
 - **Another Enrichment method:** Another method may be suitable depending on the source of your service data, for example a static data file. See [Enrichment](#) for more information.
 - **Cisco Crosswork Situation Manager UI:** An administrator can assign services to individual Situations in the UI.

Create Services and Assign Services to Situations

You can use one of the following methods, or a combination of these, to add services and assign them to Situations.

Graze API Endpoints



The **addService** endpoint enables you to create a single service or script the creation of multiple services. You can use **setSituationServices** to add one or more services to a Situation and **getSituationServices** to return a list of impacted services for a specified Situation.

See [Graze API](#) for details on the command syntax and examples. [Graze API](#)

Situation Manager Labeller

This utility allows you to create services from your custom data as it is ingested into Cisco Crosswork Situation Manager and assign those services to Situations.

See [Create Services With Situation Manager Labeller](#) for more information and an example.

Moobot

If you are using a custom Moobot to enrich on Situation creation, you can use the MoogDb **addService** and **setSituationServices** methods to create services as part of this process. See [Enrichment](#) for further information.

Another Enrichment Method

See [Enrichment](#) for further information on other enrichment methods.

Cisco Crosswork Situation Manager UI

In the UI, go into a Situation Room. Click **Services Impacted** at the top of the screen to add or remove services from the Situation. You will need administrator rights to perform this function.

[Assign Services to Teams](#)

Cisco Crosswork Situation Manager can automatically assign Situations to teams based upon the service data. You can also automatically create teams based on the service data in Situations.

See [Manage Teams](#) for details. [Manage Teams](#)

[Monitor Affected Services](#)

The Services Overview in the UI Workbench Summary allows you to view the impacted services with the highest severity Situations. You can use this information to prioritize which Situations to address first.

See [Check Impacted Services](#) for details. [Check Impacted Services](#)

[Situation Manager Labeller](#)

The Situation Manager Labeller dynamically updates Situations and performs other actions based on aggregated fields from alerts.



You can employ labeller macros in the Situation Description property of Sigalisers, for example Cookbook Recipes. The Situation Manager Moolet interprets the instructions as Cisco Crosswork Situation Manager clusters alerts into Situations.

You can use Situation Manager Labeller macros to:

- Update Situation descriptions
- Dynamically create services
- Dynamically set Situation custom info

For information on how to install and use the labeller, see:

- [Install the Situation Manager Labeller](#)
- [Create Services With Situation Manager Labeller](#)

Install the Situation Manager Labeller

You can use Situation Manager Labeller commands in Sigalisers to dynamically update Situations and perform other actions. See [Create Services With Situation Manager Labeller](#) for an example.

The utility is not included in a standard Cisco Crosswork Situation Manager installation but you can follow these instructions to download and install it.

Before You Begin

Before you install the Situation Manager Labeller, ensure you have met the following requirements:

- You have access to your Cisco Crosswork Situation Manager server and the permissions to install software and edit files.
- You have permissions to start and stop Cisco Crosswork Situation Manager services. See [Control Moogsoft AIOps Processes](#) for more information.

Install the Situation Manager Labeller

Follow these instructions to download and install the Situation Manager Labeller utility:

1. Download the Situation Manager Labeller tar file to your Cisco Crosswork Situation Manager server. Contact [Moogsoft Support](#) to obtain the latest files.

2. Extract the package. For example:

```
tar -xf SituationMgrLabeller_v7.0.0.tar
```

3. Move the existing Situation Manager Labeller file to create a backup. For example:

```
cd $MOOGSOFT_HOME/bots/moobots  
mv SituationManagerLabeller.js OLD_SituationManagerLabeller.js
```



4. Copy the extracted Situation Manager Labeller files to Cisco Crosswork Situation Manager. For example:

```
cd tmp/SituationMgrLabeller_v7.0.0
cp SituationMgrLabeller.js $MOOGSOFT_HOME/bots/moobots
cp SituationUtility.js $MOOGSOFT_HOME/contrib
cp BotUtility.js $MOOGSOFT_HOME/contrib
```

5. Edit the contents of \$MOOGSOFT_HOME/config/moolets/situation_manager.conf to set the Situation Manager Moolet to process Situation Manager Labeller output as follows:

```
name                : "SituationMgr",
classname           : "CSituationMgr",
run_on_startup      : true,
persist_state       : false,
metric_path_moolet  : false,
moobot              : "SituationMgrLabeller.js",
```

6. Restart Moogfarmd. See [Control Moogsoft AIOps Processes](#) for the commands to start, stop and restart Moogfarmd. For example:

```
service moogfarmd restart
```

Create Services Labeller

You can add Situation Manager Labeller macros to the Situation description in a Sigaliser. When Cisco Crosswork Situation Manager creates a Situation based upon the Sigaliser configuration, it automatically creates services based on the function and the alert data.

Before You Begin

Before you start to create services with Situation Manager Labeller, ensure you have met the following requirements:

- Download and install the labeller utility. See [Install the Situation Manager Labeller](#) for instructions.
- Identify the service data in your alerts or set up an Enrichment method to import the data into custom_info. See [Enrichment](#) for further information.

For example, an alert contains the following custom_info data:

```
{"services":["WinTel","Lync"]}
```

Configure a Recipe to Automatically Create Services

The following example demonstrates how to use labeller utility macros in a Cookbook Recipe to create services.

1. In the Cisco Crosswork Situation Manager UI, go to **Settings > Cookbook Recipes**.



2. Create a new Recipe, completing the mandatory fields such as name, type and clustering information. See [Configure a Recipe Manually](#) for more information.
3. Add a Situation Description that utilizes labeller macros. See the example below.
4. Activate the Recipe. See [Configure a Cookbook Recipe](#) for further information.

An example Situation description is as follows:

```
Issue affecting the service: $$UNIQUE(custom_info.services).  
$$SERVICES(custom_info.services)
```

This instructs to perform the following actions as it ingests data:

- Parse the data in the custom_info.services field of incoming events.
- Obtain unique service names for the alerts the system adds to a Situation.
- Preface the Situation's "Description" field with the text "Issue affecting the service: ".
- Create a service in the Cisco Crosswork Situation Manager database.
- Assign the service to the Situation.
- Append the Situation's Description field with the names of the impacted services.

When you are parsing a list of values stored as a JavaScript array, use \$\$ to prefix the macro. If the data is a string use the prefix \$.

Once configured, Cisco Crosswork Situation Manager processes alerts using the Situation Description and the resulting Situation appears in the Cisco Crosswork Situation Manager UI.

For example:

<input type="checkbox"/>	SEVERITY	ID ↓	OWNED BY	TEAMS	DESCRIPTION	SERVICES IMPACTED	TOTAL ALERTS
<input type="checkbox"/>	Critical	#48	Administrator	Cloud DevOps	Issue affecting the service: ["WinTel","Lync"].	WinTel, Lync	4

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: ["Cookbook", "Tempus"],  
#     alert_threshold      : 3,  
#     sig_similarity_limit : 0.75  
# }
```

This merge group would only merge Situations created by the Cookbook 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).

Monitor and Troubleshoot Cisco Crosswork Situation Manager

This document details the available health and performance indicators included with the Cisco Crosswork Situation Manager system. It also provides some guidance on how to monitor your system and how to troubleshoot performance problems.

Processing Metrics

Navigate to **System Settings > Self Monitoring > Processing Metrics** to see a breakdown of the current state of the system based on the metrics received from the running components.

- The Moogfarmd process and all LAMs publish detailed performance information.
- A bullet chart at the top of the page shows the key performance metric for the system: Current Maximum Event Processing Time. The defined performance ranges are color coded: good (green), marginal (yellow) and bad (red). As the metric changes the bullet chart updates to reflect good, marginal or bad performance.
- The system calculates Current Maximum Event Processing Time as the approximate 95th percentile of the current maximum time in seconds that it takes for an event to make its way through the system from its arrival at a LAM until its final processing by a Moollet in Moogfarmd.
- By default, AlertBuilder, AlertRulesEngine and All Sigalisers are used to calculate the Current Maximum Event Processing Time metric.



- You can configure the `metric_path_moolet` property in `moog_farmd.conf` to specify the Moolets to use to calculate Current Maximum Event Processing Time.
- By default, the good, marginal and bad ranges of the bullet chart are set to 0-10secs, 10-15secs and 15-20secs respectively. You can change the configuration in the `eventProcessingTimes` section in the `portal` block of `$MOOGSOFT_HOME/ui/html/web.conf`.

Good performance means LAMs are consuming and publishing events without problem as indicated by:

- Message Queue Size is 0.
- Socket Backlog (if relevant) is not increasing.

Additionally, Moogfarmd is consuming and processing events successfully as indicated by all of:

- Total Abandoned Messages is 0 for the majority of the time.
- Asynchronous Task Queue Size is 0 for the majority of the time.
- Cookbook Resolution Queue is 0 for the majority of the time.
- Message backlogs for all Moolets is 0 for the majority of the time.
- The Messages Processed count for all running Moolets should be the same (unless custom configuration causes event routing through different Moolets) i.e. no Moolet is falling behind.

The above should lead to a stable low Current Maximum Event Processing Time depending on the complexity of the system.

Marginal or **Bad** performance means LAMs are not consuming and publishing events at the rate at which they receive them, as indicated by:

- Message Queue Size is > 0 and likely increasing.
- Socket Backlog is increasing.

Additionally, Moogfarmd is not consuming and processing events in a timely fashion as indicated by some or all of:

- Total Abandoned Messages is constantly > 0 and likely increasing.
- Asynchronous Task Queue Size is > 0 and likely increasing.
- Cookbook Resolution Queue is constantly > 0 and likely increasing.
- Message backlogs for all Moolets is constantly > 0 and likely increasing.



- The Messages Processed count for all running Moolets is not the same indicating that some Moolets are falling behind. This doesn't apply for cases where custom configuration causes event routing through different Moolets.

The above will likely lead to an unstable high Current Maximum Event Processing Time depending on the complexity of the system.

See [Self Monitoring](#) for more detail.

[Graze getSystemStatus Endpoint](#)

The getSystemStatus endpoint returns useful information about running processes within the system. For example:

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

See [Graze API](#) for more detail.

[Moogfarmd Health Logging](#)

Moogfarmd writes detailed health information in JSON format to its log file once a minute. Information falls into five logical blocks:

- totals: running totals since Moogfarmd was started.
- interval_totals: running totals since the last 60 second interval)
- current_state: a snapshot of the important queues in Moogfarmd
- garbage_collection: JVM garbage collection data
- JVM_memory: JVM memory usage data
- message_queues: Queue usage and capacity

Example output:

```
WARN : [HLog ][20180510 20:39:55.538 +0100] [CFarmdHealth.java]:533
+|{"garbage_collection":{"total_collections_time":12827,"last_minute_collec
tions":0,"last_minute_collections_time":0,"total_collections":1244},"curren
t_state":{"pending_changed_situations":0,"total_in_memory_situations":4764,
"situations_for_resolution":0,"event_processing_metric":0.04747474747474747
5,"message_queues":{"AlertBuilder":0,"TeamsMgr":0,"Housekeeper":0,"Indexer"
:0,"bus_thread_pool":0,"Cookbook3":0,"Cookbook1":0,"SituationMgr":0,"Situat
ionRootCause":0,"Cookbook2":0},"in_memory_entropies":452283,"cookbook_resol
ution_queue":0,"total_in_memory_priority_situations":0,"active_async_tasks_
count":0},"interval_totals":{"created_events":1782,"created_priority_situat
ions":0,"created_external_situations":0,"created_situations":10,"messages_p
rocessed":{"TeamsMgr":182,"Housekeeper":0,"AlertBuilder":1782,"Indexer":208
2,"Cookbook3":1782,"SituationRootCause":172,"Cookbook1":1782,"SituationMgr"
:172,"Cookbook2":1782},"alerts_added_to_priority_situations":0,"alerts_adde
d_to_situations":111,"situation_db_update_failure":0},"JVM_memory":{"heap_u
sed":1843627096,"heap_committed":3007840256,"heap_init":2113929216,"nonheap
```



```
_committed":66912256,"heap_max":28631367680,"nonheap_init":2555904,"nonheap_
_used":64159032,"nonheap_max":-
1},"totals":{"created_events":453252,"created_priority_situations":0,"creat
ed_external_situations":0,"created_situations":4764,"alerts_added_to_priori
ty_situations":0,"alerts_added_to_situations":36020,"situation_db_update_fa
ilure":0}}|+
```

The message_queues block contains string values and queue limits. "-" represents an unlimited queue. An example message_queues block is as follows:

```
"message_queues":{"AlertBuilder":"0/-","Cookbook":"0/-","Housekeeper":"0/-
","Indexer":"0/-","bus_thread_pool":"0/-","SituationMgr":"0/-"}
```

In a healthy system that is processing data:

- The count of created events and created Situations should increase.
- The messages_processed should show that Moolets are processing messages.
- The current_state.message_queues should not be accumulating (there may be spikes).
- The total_in_memory Situations should increase over time but will reduce periodically due to the retention_period.
- The situation_db_update_failure should be zero.

Tomcat Servlet Logging

Tomcat writes counter information from each of the main servlets to its catalina.out once a minute.

Example output:

```
WARN : [Thread-][20180510 20:57:05.501 +0100] [CReporterThread.java]:136
+|MoogPoller read [16722] MooMs messages in the last [60] seconds.|+
WARN : [Thread-][20180510 20:57:07.169 +0100] [CReporterThread.java]:136
+|Graze handled [55] requests in the last [60] seconds.|+
WARN : [Thread-][20180510 20:57:10.181 +0100] [CReporterThread.java]:136
+|MoogSvr handled [86] requests in the last [60] seconds.|+
WARN : [Thread-][20180510 20:58:03.197 +0100] [CReporterThread.java]:136
+|Situations similarity component calculated similarities for [264]
situations in the last [60] seconds.|+
```

The counters are:

- Number of MoogSvr requests in the last minute (i.e. number of standard UI requests made).
- Number of Moogpoller MooMs messages in the last minute (i.e. number of messages read from the bus).
- Number of Graze requests in the last minute.



- Number of similar Situations calculated in the last minute.

In a healthy system that is processing data:

- The Moogpoller count should always be non-zero.
- The MoogSvr and Graze counters may be zero, but should reflect the amount of UI and Graze activity.
- The similar Situations counter may be zero but should reflect the number of similar Situations that are occurring in the system.

Database Pool Diagnostics

Cisco Crosswork Situation Manager features a capability to print out the current state of the DBPool in both Moogfarmd and Tomcat. This can be very useful to diagnose problems with slow event processing or UI response.

To trigger logging, run the `ha_cntl` utility and pass the cluster name using the `-i` argument. For example:

```
ha_cntl -i MOO
```

This will perform task "diagnostics" all groups within the [MOO] cluster. Diagnostics results will be in the target process log file.

Are you sure you want to continue? (y/N)

The utility triggers logging to `/var/log/moogsoft/moogfarmd.log`. For example in a performant system:

```
WARN : [pool-1-][20180511 10:06:07.690 +0100] [CDbPool.java]:792 +|[farmd]
DATABASE POOL DIAGNOSTICS:|+
WARN : [pool-1-][20180511 10:06:07.690 +0100] [CDbPool.java]:793 +|[farmd]
Pool created at [20180510 17:54:48.911 +0100].|+
WARN : [pool-1-][20180511 10:06:07.690 +0100] [CDbPool.java]:797 +|[farmd]
[2] invalid connections have been removed during the lifetime of the
pool.|+
WARN : [pool-1-][20180511 10:06:07.690 +0100] [CDbPool.java]:833 +|[farmd]
Pool size is [10] with [10] available connections and [0] busy.|+
```

It also triggers logging to `/usr/share/apache-tomcat/logs/catalina.out`. For example:

```
WARN : [0:MooMS][20180511 10:06:07.690 +0100] [CDbPool.java]:792
+|[SituationSimilarity] DATABASE POOL DIAGNOSTICS:|+
WARN : [0:MooMS][20180511 10:06:07.690 +0100] [CDbPool.java]:793
+|[SituationSimilarity] Pool created at [20180510 17:55:04.262 +0100].|+
WARN : [3:MooMS][20180511 10:06:07.690 +0100] [CDbPool.java]:792
+|[MoogPoller] DATABASE POOL DIAGNOSTICS:|+
WARN : [3:MooMS][20180511 10:06:07.690 +0100] [CDbPool.java]:793
+|[MoogPoller] Pool created at [20180510 17:55:01.990 +0100].|+
WARN : [0:MooMS][20180511 10:06:07.690 +0100] [CDbPool.java]:833
+|[SituationSimilarity] Pool size is [5] with [5] available connections and
[0] busy.|+
```



```
WARN : [3:MooMS][20180511 10:06:07.691 +0100] [CDbPool.java]:833
+|[MoogPoller] Pool size is [10] with [10] available connections and [0]
busy.|+
WARN : [1:MooMS][20180511 10:06:07.693 +0100] [CDbPool.java]:792
+|[ToolRunner] DATABASE POOL DIAGNOSTICS:|+
WARN : [1:MooMS][20180511 10:06:07.694 +0100] [CDbPool.java]:793
+|[ToolRunner] Pool created at [20180510 17:55:00.183 +0100].|+
WARN : [1:MooMS][20180511 10:06:07.694 +0100] [CDbPool.java]:792 +|[MoogSvr
: priority] DATABASE POOL DIAGNOSTICS:|+
WARN : [1:MooMS][20180511 10:06:07.694 +0100] [CDbPool.java]:833
+|[ToolRunner] Pool size is [5] with [5] available connections and [0]
busy.|+
WARN : [1:MooMS][20180511 10:06:07.694 +0100] [CDbPool.java]:793 +|[MoogSvr
: priority] Pool created at [20180510 17:54:56.800 +0100].|+
WARN : [1:MooMS][20180511 10:06:07.695 +0100] [CDbPool.java]:797 +|[MoogSvr
: priority] [5] invalid connections have been removed during the lifetime
of the pool.|+
WARN : [1:MooMS][20180511 10:06:07.695 +0100] [CDbPool.java]:833 +|[MoogSvr
: priority] Pool size is [25] with [25] available connections and [0]
busy.|+
WARN : [1:MooMS][20180511 10:06:07.695 +0100] [CDbPool.java]:792 +|[MoogSvr
: normal priority] DATABASE POOL DIAGNOSTICS:|+
WARN : [1:MooMS][20180511 10:06:07.695 +0100] [CDbPool.java]:793 +|[MoogSvr
: normal priority] Pool created at [20180510 17:54:56.877 +0100].|+
WARN : [1:MooMS][20180511 10:06:07.695 +0100] [CDbPool.java]:833 +|[MoogSvr
: normal priority] Pool size is [50] with [50] available connections and
[0] busy.|+
```

In both of these examples, the connections are "available" and none show as busy. However, in a busy system with flagging performance, the Moogfarmd log will show different results. In the example below, all connections are busy and have been held for a long time. This type of critical issue causes Moogfarmd to stop processing:

```
WARN : [pool-1-]20180309 16:49:30.031 +0000] [CDbPool.java]:827 +|[farmd]
Pool size is [10] with [0] available connections and [10] busy.|+
WARN : [pool-1-]20180309 16:49:30.031 +0000] [CDbPool.java]:831 +|[The busy
connections are as follows:
1: Held by 5:SituationMgrLOGFILECOOKBOOK for 173603 milliseconds. Checked
out at [CArchiveConfig.java]:283.
2: Held by 7:SituationMgrSYSLOGCOOKBOOK for 173574 milliseconds. Checked
out at [CArchiveConfig.java]:283.
3: Held by 8:SituationMgrSYSLOGCOOKBOOK for 173658 milliseconds. Checked
out at [CArchiveConfig.java]:283.
4: Held by 9:SituationMgrSYSLOGCOOKBOOK for 173477 milliseconds. Checked
out at [CArchiveConfig.java]:283.
5: Held by 8:TeamsMgr for 173614 milliseconds. Checked out at
[CArchiveConfig.java]:283.
6: Held by 4:SituationMgrSYSLOGCOOKBOOK for 173514 milliseconds. Checked
out at [CArchiveConfig.java]:283.
7: Held by 5:PRC Request Assign - SituationRootCause for 173485
milliseconds. Checked out at [CArchiveConfig.java]:283.
```



```
8: Held by 2:SituationMgrSYSLOGCOOKBOOK for 173661 milliseconds. Checked
out at [CArchiveConfig.java]:283.
9: Held by 6:SituationMgrSYSLOGCOOKBOOK for 173631 milliseconds. Checked
out at [CArchiveConfig.java]:283.
10: Held by 6:TeamsMgr for 172661 milliseconds. Checked out at
[CArchiveConfig.java]:283. |+
```

It is expected that occasionally some of the connections will be busy but as long as they are not held for long periods of time then the system will be functioning normally.

You can use the following bash script to automatically gather DBPool diagnostics:

```
#!/bin/bash

#Get the cluster name
CLUSTER=$(($MOOGSOFT_HOME/bin/utils/moog_config_reader -k ha.cluster)

#Get the current line numbers of latest log lines
FARMLINES=$(wc -l /var/log/moogsoft/moogfarmd.log|awk '{print $1}')
TOMLINES=$(wc -l /usr/share/apache-tomcat/logs/catalina.out|awk '{print
$1}')

#Run ha_cntl -i <cluster>
ha_cntl -i $CLUSTER -y > /dev/null

sleep 5

#Print the results
echo "moog_farmd:"
tail -n +$FARMLINES /var/log/moogsoft/moogfarmd.log|egrep "CDbPool|Held by"

echo "tomcat:"
tail -n +$TOMLINES /usr/share/apache-tomcat/logs/catalina.out|egrep
"CDbPool|Held by"
```

To run the script, execute the following command:

```
./get_dbpool_diag.sh
```

[MySQL Slow Query Logging](#)

Slow query logging captures long running queries that are impacting the database. You can enable the feature as follows:

1. Check the current settings in MySQL for the feature:

```
mysql> show variables like '%slow_query_log%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| slow_query_log | OFF   |
| slow_query_log_file | /var/log/mysql-slow.log |
```



```
+-----+
2 rows in set (0.00 sec)
```

```
mysql> show variables like 'long_query%';
```

```
+-----+
| Variable_name | Value      |
+-----+
| long_query_time | 10.000000 |
+-----+
1 row in set (0.00 sec)
```

2. Ensure that the file specified in the `slow_query_log_file` setting exists and has the correct permissions. If not create it/set permissions :

```
touch /var/log/mysql-slow.log
chown mysql:mysql /var/log/mysql-slow.log
```

3. Enable the slow query log:

```
mysql> set global slow_query_log=on;
```

After that, queries that take longer than 10 seconds to execute will appear in the log file. For example:

```
/usr/sbin/mysqld, Version: 5.7.19 (MySQL Community Server (GPL)).
started with:
Tcp port: 3306 Unix socket: /var/lib/mysql/mysql.sock
Time Id Command Argument
# Time: 2018-02-02T19:12:20.822756Z
# User@Host: ermintrude[ermintrude] @ localhost [127.0.0.1] Id: 98
# Query_time: 55.161516 Lock_time: 0.000025 Rows_sent: 1 Rows_examined:
25878591
use moogdb;
SET timestamp=1517598740;
SELECT COALESCE(MIN(GREATEST(last_state_change,last_event_time)),
UNIX_TIMESTAMP(SYSDATE())) as oldest FROM alerts WHERE alerts.alert_id
NOT IN (SELECT sig_alerts.alert_id FROM sig_alerts);
# Time: 2018-02-02T19:13:19.255277Z
# User@Host: ermintrude[ermintrude] @ localhost [127.0.0.1] Id: 98
# Query_time: 56.131108 Lock_time: 0.000028 Rows_sent: 515
Rows_examined: 25878591
SET timestamp=1517598799;
SELECT alerts.alert_id FROM alerts WHERE alerts.alert_id NOT IN (SELECT
sig_alerts.alert_id FROM sig_alerts) AND
GREATEST(last_state_change,last_event_time) BETWEEN 1417547486 AND
1417633885;
```

The value of the `long_query_time` setting can also be adjusted up or down as suits.



RabbitMQ Admin UI

RabbitMQ includes an admin UI that gives performance information about the message bus. By default this is accessible via `http://<hostname>:15672` with credentials `moogsoft/m00gs0ft`. Check for the following scenarios:

- Early warning of any system resource issues in the "Nodes" section on the Overview page. For example, file/socket descriptors, Erlang processes, memory and disk space.
- Build-up of "Ready" messages in a queue - this indicates a message queue is forming. This means that the associated Cisco Crosswork Situation Manager process is not consuming messages from this queue. It could also point to an orphaned queue that no longer has an associated consumer. This could happen if "message_persistence" has been enabled in `system.conf` and `Moogfarmd` and or `Tomcat` has been reconfigured with a different HA process group name.

See the [RabbitMQ docs](#) for information on how to use the admin UI.

Other Utilities

MySQLTuner provides useful diagnostics and recommendations on MySQL settings. See [MySQLTuner](#) for more information.

To monitor the CPU and memory usage of the running components of a Cisco Crosswork Situation Manager system, you can run the following script that offers simple CPU and memory monitoring of the RabbitMQ, Socket LAM, Moogfarmd, Tomcat and MySQL processes:

```
#!/bin/bash

SLEEPTIME=$1

f_return_metrics() {
    PROCID=$1
    TOPOUTPUT=`top -p $PROCID -n1 | tail -2 | head -1 | sed 's/[^
]\+s\(.*\)/\1/g'`
    PROCICPU=`echo $TOPOUTPUT | awk '{print $8}'`
    if [ "$PROCICPU" == "S" ]; then PROCICPU=`echo $TOPOUTPUT | awk
'{print $9}'`;fi
    PROCPCPU=`ps -p $PROCID -o pcpu|tail -1|awk '{print $1}'`
    PROCMEM=`ps -p $PROCID -o rss|tail -1|awk '{print $1}'`
    echo $PROCICPU,$PROCPCPU,$PROCMEM
}

#Capture PIDs
RABBITPID=`ps -ef|grep beam|grep -v grep|awk '{print $2}'`
LAMPID=`ps -ef|grep socket_lam|grep java|grep -v grep|awk '{print $2}'`
MYSQLPID=`ps -ef|grep mysqld|grep -v mysqld_safe|grep -v grep|awk '{print`
```



```
$2}``
TOMCATPID=`ps -ef|grep tomcat|grep java|grep -v grep|awk '{print $2}``
FARMDPID=`ps -ef|grep moog_farmd|grep java|grep -v grep|awk '{print $2}``

echo
"DATE,TIME,RABBITICPU(%),RABBITPCPU(%),RABBITRSS(Kb),LAMICPU(%),LAMP CPU(%),
LAMRSS(Kb),FARMDICPU(%),FARMDPCPU(%),FARMDRSS(Kb),TOMCATICPU(%),TOMCATPCPU(
%),TOMCATRSS(Kb),MYSQLICPU(%),MYSQLPCPU(%),MYSQLRSS(Kb)"

while [ true ]; do

    DATENOW=`date +"%m-%d-%y"`
    TIMENOW=`date +"%T"`

    RABBITMEAS=$(f_return_metrics $RABBITPID)
    LAMMEAS=$(f_return_metrics $LAMPID)
    FARMDMEAS=$(f_return_metrics $FARMDPID)
    TOMCATMEAS=$(f_return_metrics $TOMCATPID)
    MYSQLMEAS=$(f_return_metrics $MYSQLPID)
    TOMCATMEAS=$(f_return_metrics $TOMCATPID)

    echo
"$DATENOW,$TIMENOW,$RABBITMEAS,$LAMMEAS,$FARMDMEAS,$TOMCATMEAS,$MYSQLMEAS"

    sleep $SLEEPTIME

done
```

Example usage and output:

```
[root@ldev04 640]# ./perfmon.sh 5
DATE,TIME,RABBITICPU(%),RABBITPCPU(%),RABBITRSS(Kb),LAMICPU(%),LAMP CPU(%),L
AMRSS(Kb),FARMDICPU(%),FARMDPCPU(%),FARMDRSS(Kb),TOMCATICPU(%),TOMCATPCPU(
%),TOMCATRSS(Kb),MYSQLICPU(%),MYSQLPCPU(%),MYSQLRSS(Kb)
05-10-
18,22:44:26,28.0,8.5,203068,2.0,1.0,557092,20.0,13.5,2853408,4.0,2.1,568058
4,28.0,17.4,9657152
05-10-
18,22:44:34,14.0,8.5,183492,4.0,1.0,557092,16.0,13.5,2850484,0.0,2.1,568058
4,33.9,17.4,9657152
05-10-
18,22:44:43,0.0,8.5,181072,0.0,1.0,557092,0.0,13.5,2850484,0.0,2.1,5680584,
4.0,17.4,9658312
05-10-
18,22:44:51,12.0,8.5,181040,0.0,1.0,557092,0.0,13.5,2850484,0.0,2.1,5680584
,4.0,17.4,9658312
05-10-
18,22:44:59,0.0,8.5,181040,0.0,1.0,557092,0.0,13.4,2850484,0.0,2.1,5680584,
0.0,17.4,9658312
```

Notes:



- Script only outputs to the console so should be redirected to a file for logging results
- Output is in csv format.
- ICPU = "Instantaneous CPU Usage (%)"
- PCPU = "Percentage of CPU usage since process startup (%)"
- RSS = "Resident Set Size i.e. Memory Usage in Kb"
- For CPU measurements a measure of 100% represents all of one processor so results > 100% are achievable for multi-threaded processes.

Troubleshooting Performance Problems

If the system is showing signs of latency in alert or Situation creation then the problem is likely with Moogfarmd and/or the database. The following diagnostic steps will help you track down the cause:

Step	Description	Possible Cause and Resolution
1	Check the Moogfarmd log for any obvious errors or warning.	Cause may be evident from any warnings or errors.
2	Check the Self Monitoring > Processing Metrics Page	<p>If the event_process_metric is large and/or increasing then something is backing up.</p> <p>Check Moogfarmd health logging also for sign of message_queue build-up in any of the Moolets.</p>
3	Check the CPU/memory usage of the server itself.	If the server, as a whole, is running close to CPU or memory limit and no other issues can be found (e.g. rogue processes or memory leaks in the Cisco Crosswork Situation Manager components) then consider adding more resource to the server or distributing the Cisco Crosswork Situation Manager components.
4	Check whether the Moogfarmd java process is showing constant high CPU/memory usage.	<p>Moogfarmd may be processing an event or Situation storm.</p> <p>Check Moogfarmd health logging also for sign of message_queue build-up in any of the Moolets. Backlog should clear assuming storm subsides.</p>

- 5 Has the memory of the Moogfarmd java processed reached a plateau?

Moogfarmd may have reached its java heap limit. Check the -Xmx settings of Moogfarmd. If not specified has Moogfarmd reached approximately a quarter of the RAM on the server? Increase the -Xmx settings as appropriate and restart the Moogfarmd service.
- 6 Is the database tuned?

Check the innodb-buffer-pool-size and innodb_buffer_pool_instances settings in /etc/my.cnf as per Tuning section above. Ensure they are set appropriately and restart mysql if changes are made.
- 7 Check the server for any other high CPU or memory processes or that which might be impacting the database.

Something may be hogging CPU/memory on the server and starving Moogfarmd of resources.

The events_analyser utility may be running or a sudden burst of UI or Graze activity may be putting pressure on the database and affecting Moogfarmd.
- 8 Run DBPool Diagnostics (see previous section) several times to assess current state of Moogfarmd to database connections.

Moogfarmd database connections may be maxed out with long running connections - this may indicate a processing deadlock - perform a kill - 3 <pid> on the Moogfarmd java process to generate a thread dump (in the Moogfarmd log) and send it to Moogsoft Support.

Alternatively Moogfarmd may be very busy with lots of short but frequent connections to the database. Consider increasing the number DBPool connections for Moogfarmd by increasing the top-level "threads" property in the Moogfarmd configuration file and restarting the Moogfarmd service.
- 9 Turn on MySQL slow query logging (see earlier section on how to do this)

Slow queries from a Moobot in Moogfarmd may be causing problems and they should be reviewed for

efficiency.

Alternatively slow queries from other parts of the system may be causing problems (e.g. nasty UI filters).

Slow queries may also be down to the sheer amount of data in the system. Consider enabling Database Split to move old data and/or using the Archiver to remove old data.

- 10 Check Moogfarmd Situation resolution logging using:

```
grep "Resolve has been running  
for"  
/var/log/moogsoft/moogfarmd.log
```

If this logging shows non-zero upward trend in "Resolve" time then Moogfarmd is struggling with the number of "in memory" Situations for its calculations.

Check the Moogfarmd health logging for the current count of "in memory" situations and consider reducing the retention_period setting in the Moogfarmd log (will need a Moogfarmd restart) and/or closing more old Situations.

- 11 Is Moogfarmd memory constantly growing over time and a memory leak is suspected?

Note that Moogfarmd memory does typically increase for periods of time then is trimmed back via Java garbage collection and Sigaliser memory purge (via retention_period property).

Take periodic heap dumps from the Moogfarmd java process and send them to Moogsoft Support so they can analyse the growth. Use the following commands:

```
DUMPFIL=/tmp/farmd-heapdump-  
$(date +%s).bin  
sudo -u moogsoft jmap -  
dump:format=b,file=$DUMPFIL $(ps  
-ef|grep java|grep moog_farmd|awk  
'{print $2}')
```

Notes:

- jmap needs java jdk to be installed. "yum install jdk" should suffice to install this.
- generating a heap dump is like to make the target process very busy for a period of time and also triggers a garbage collection so the



memory usage of the process may well reduce.

- heapdump files may be very large.

If the system is showing signs of slow UI performance, such as long login times, spinning summary counters, or other, then the problem is likely with Tomcat and/or the database. The following diagnostic steps will help you track down the cause:

Step	Description	Possible Cause and Resolution
1	Check catalina.out for any obvious errors or warning.	Cause may be evident from any warnings or errors.
2	Check browser console or any errors or timing out requests.	Possibly a bug or more likely that the query to the database associated with the request is taking longer than 30secs (the default browser timeout). Root cause of this should be investigated.
3	Check network latency between browser client machine and server using ping.	Latency of ≥ 100 ms can make login noticeably slower.
4	Check the CPU/memory usage of the server itself.	If the server, as a whole, is running close to CPU or memory limit and no other issues can be found (e.g. rogue processes or memory leaks in the Cisco Crosswork Situation Manager components) then consider adding more resource to the server or distributing the Cisco Crosswork Situation Manager components.
5	Check MoogSvr/Moogpoller/Graze counter logging in catalina.out	Tomcat may be processing a high number of requests or bus updates. If Moogpoller count is zero then something may be wrong with Tomcat > RabbitMQ connection. Check RabbitMQ admin UI for signs of message queue build-up.
6	Check whether Tomcat java process is showing constant high CPU/memory usage.	Tomcat may be processing the updates from an event or situation storm. Backlog should clear assuming storm subsides.
7	Has the memory of the Tomcat java process reached a plateau?	Tomcat may have reached its java heap limit. Check the -Xmx setting in /etc/init.d/apache-tomcat. Increase the -Xmx settings as appropriate and

- restart the apache-tomcat service.
- 8 Is the database tuned?
Check the **innodb-buffer-pool-size** and **innodb_buffer_pool_instances** settings in /etc/my.cnf as per Tuning section above. Ensure they are set appropriately and restart mysql if changes are made.
- 9 Check the server for any other high CPU or memory processes or that which might be impacting the database.
Something may be hogging CPU/memory on the server and starving Tomcat of resources.
The Events Analyser utility may be running or a sudden burst of Moogfarmd or Graze activity may be putting pressure on the database and affecting the UI.
- 10 Run DBPool Diagnostics (see previous section) several times to assess current state of Tomcat > Database connections.
Tomcat database connections may be maxed out with long running connections - this may indicate a processing deadlock - perform a `kill -3 <pid>` on the Tomcat java process to generate a thread dump (in catalina.out) and send it to Cisco Crosswork Situation Manager Support.

Alternatively Tomcat may be very busy with lots of short but frequent connections to the database. A Graze request bombardment is another possibility (Graze does not currently have a separate DB Pool). Consider increasing the number DBPool connections for Tomcat by increasing the related properties in `servlets.conf` and restarting the apache-tomcat service.
- 11 Turn on MySQL slow query logging (see earlier section on how to do this)
Slow queries from nasty filters in the UI may be causing problems and they should be reviewed for efficiency.

Alternatively slow queries from other parts of the system may be causing problems (e.g. inefficient Moobot code).

Slow queries may also be down to the sheer amount of data in the system. Consider enabling **Database Split** to move old data and/or using the **Archiver** to remove old data.
- 12 Is Tomcat memory constantly growing over time and a
Take periodic heap dumps from the Tomcat java process and send them to Cisco support so they can analyse the growth. Use the



memory leak is suspected?

Note that Tomcat memory does typically increase for periods of time then is trimmed back via java garbage collection.

following commands:

```
DUMPFIL=/tmp/tomcat-heapdump-$(date +%s).bin
sudo -u tomcat jmap -
dump:format=b,file=$DUMPFIL $(ps -
ef|grep java|grep tomcat|awk '{print
$2}')
bzip2 $DUMPFIL
```

Notes:

- jmap needs Java JDK to be installed. "yum install jdk" should suffice to install this.
- generating a heap dump is likely to make the target process very busy for a period of time and also triggers a garbage collection so the memory usage of the process may well reduce.
- heapdump files may be very large

Cisco Crosswork Situation ManagerComponent Performance

Cisco Crosswork Situation Manager features the ability to ingest large amounts of event data from various sources, process the data using configurable logic, and display the data to multiple concurrent users. This document outlines the various system components and how their interactions can impact system performance. It includes performance tuning suggestions where applicable.

To learn about opportunities to plan your implementation for increased performance capabilities, see [Scaling Your Moogsoft AIOps Implementation](#).

For information on monitoring your system performance and handling performance issues, see [Monitor and Troubleshoot Moogsoft AIOps](#).

System Component Summary

Cisco Crosswork Situation Manager comprises several components which have tuning and configuration options available:

- Integrations and LAMs that listen or poll for data, parse and encode them into discrete events, and then pass the events to the Message Bus.
- The Message Bus (RabbitMQ) that receives published messages from integrations and LAMs. It publishes messages destined for data processing (Moogfarmd) and the web application server.
- The system datastore (MySQL) that handles transactional data from other parts of the system: integrations and LAMs, data processing, and the web application server.



- The data processing component (Moogfarmd), an application that consumes messages from the Message Bus. It processes event data in a series of servlet-like modules called Moolets. Moogfarmd reads and writes to the database and publishes messages to the bus.
- The web application server (Apache Tomcat) that reads and writes to the bus and the database.

The diagram below shows the general data flow of the components:



Other components include:

- A proxy (Nginx) for the web application server and for integrations. See the [Nginx docs](#) for more information.
- The search engine (Elasticsearch) for the UI that indexes documents from the indexer Moolet in the data processing series. It returns search results to Apache Tomcat. See the [Elasticsearch documentation](#) for more information.

Integration Performance

Event data enters the system via integrations and LAMs. Integrations and LAMs running on a large system can normally process up to 10,000 events per second and publish them to the Message Bus at an equivalent rate. Integrations can buffer events under event storm conditions. The following factors affect the capacity to process events:

- CPU clock speed and number of available cores.
- Threads setting.
- Complexity of the LAMbot logic.
- Whether you have enabled "guaranteed delivery settings". For example, `rest_response_mode` for the REST LAM.
- Log level. For example, DEBUG is the slowest.

You can specify a value for the number of threads in the LAM's configuration file to control the number of active threads for the integration. To tune the socket LAM, for example, edit `socket_lam.conf`. Increasing the number of threads can improve ingestion performance. However it will also result in higher CPU usage and may cause internal buffering. Buffering increases memory usage until the buffer is cleared.



Message Bus Performance

RabbitMQ is very lightweight and, in all known cases, has been able to process the incoming event rate from Integrations. Refer to the [RabbitMQ documentation](#) for its performance tuning options.

Database Performance

Manage and tune your MySQL instance as you would any other database system in your enterprise. In addition to the standard tuning options in the [MySQL documentation](#), consider the following recommendations for settings in `/etc/my.cnf` :

- On servers with ≥ 16 GB RAM that run MySQL and Cisco Crosswork Situation Manager applications ,set `innodb-buffer-pool-size` to 50% of system RAM.
- On servers where only MySQL is running, set `innodb-buffer-pool-size` to 80% of system RAM.
- If `innodb-buffer-pool-size` > 8 GB, increase the `innodb-buffer-pool-instances` to divide the buffer buffer pool into 1 G (GB) chunks to the maximum supported value of 64 G. For example, if your buffer pool size is 64 GB:

```
innodb-buffer-pool-size=64G
innodb_buffer_pool_instances=64
```

Data Processing Performance

The data processing component for Cisco Crosswork Situation Manager, Moogfarmd, is the most complex and configurable system component. It offers a range of performance capabilities depending on which Moolets you use and the workload for those Moolets. The following factors affect Moogfarmd performance:

- Incoming event rate from integrations and LAMs.
- CPU clock speed and number of available cores.
- Available memory and `-Xmx` setting of Moogfarmd process.
- Top-level and per-Moolet threads setting.
- Number of Moolets and their complexity and/or interaction with external services.
- Database load from other parts of the system, for example API requests.
- Incoming messages from the Message Bus or other parts of the system.

Moogfarmd buffers messages in message storm conditions. Each Moolet has its own message queue that enables it to handle message storms and process the backlog once the storm has cleared.

You can configure thread allocation for Moogfarmd in the `$MOOGSOFT_HOME/config/moog_farmd.conf` file as follows:

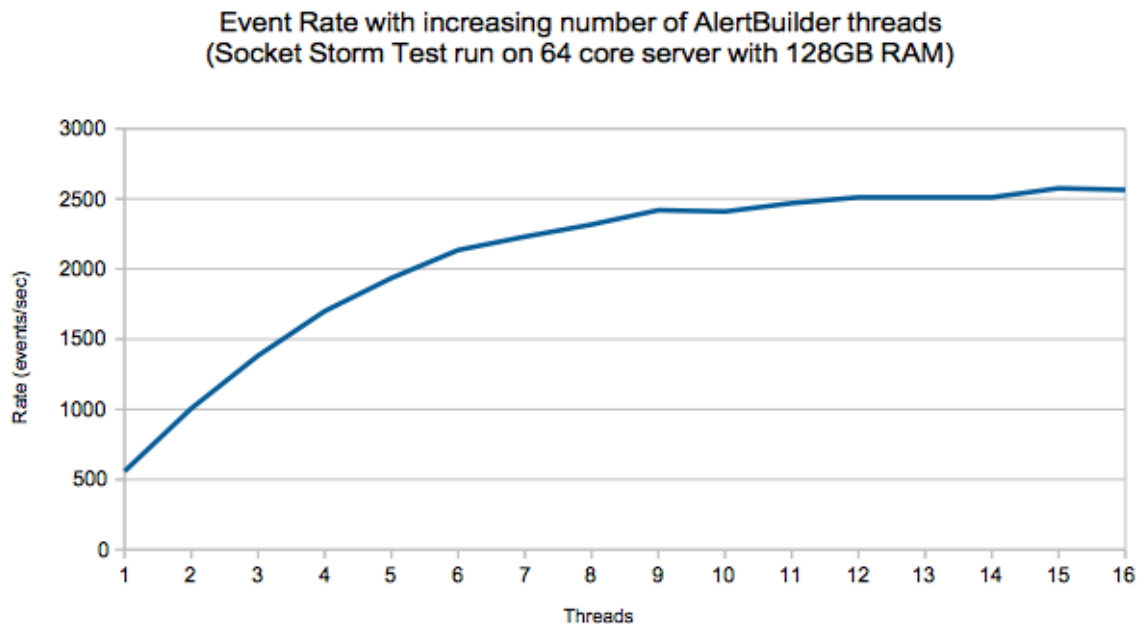


- The top-level threads property controls the following:
 - The default number of threads for each Moolet unless you specify a setting for a particular Moolet.
 - The size of the database pool for Moogfarmd to database connections.
- The per-Moolet level threads property allows individual control of the number of threads for a particular Moolet.

Increasing either setting can lead to improved processing performance but will likely increase CPU and memory usage. Too many threads can lead to overload of connections or transactions to the database and impact other areas of the system. For example, increasing the number of threads for the Alert Builder Moolet can improve the event processing rate, but increases load on the database potentially causing deadlocks.

Alert Builder Moolet

The main performance gateway for Moogfarmd is the Alert Builder because it interacts with MySQL the most. In simple configurations with a tuned MySQL database and no other load, you can increase number of threads for the Alert Builder to process up to 2500 events per second and write them to the database at an equivalent rate. The following graph illustrates the performance impact of adding Alert Builder threads for moogfarmd running only with Alert Builder.



This scenario does not account for other database load, other Moolets, or any custom logic added to the Alert Builder Moobot. Event processing would run at about half this rate in a real-world case.

Signalisers



Cisco Crosswork Situation Manager clustering algorithms or Sigalisers, employ complex calculations. Depending on its settings, the Sigaliser can account for a lot of processing time and memory within Moogfarmd. It is impossible to predict a processing rate for these algorithms because as they vary greatly according to configuration and workload. Normally Sigalisers do not add much load to the database except in a burst of Situation creation. Moogfarmd retains previously created active Situations in memory according to the `retention_period` setting in the Moogfarmd configuration file. You can expect memory to grow in Moogfarmd as a consequence under a high rate of Situation generation.

Other Moolets

The performance of other Moolets vary based upon configuration and the rate at which they receive messages to process. Moolets that interact with external services may introduce processing delay to Moogfarmd when there is network or processing latency associated with the external service.

Web Application Server Performance

The Apache Tomcat servlets provide the backend for the Cisco Crosswork Situation ManagerUI which drives the end-user experience. Scalability tests show that a single Tomcat instance can support up to 500 concurrent UI users before response times degrade. Tomcat performance depends on the following factors:

- Incoming event rate from integrations and LAMs.
- Incoming messages from other parts of the system, such as Moogfarmd.
- CPU clock speed and number of available cores.
- Available memory and `-Xmx` setting of the Tomcat process.
- Database load from other parts of the system.
- Number and complexity of alert and Situation filters being used.
- Activities of the users.

To provide quicker load times for users, the UI employs caching benefits for filtered views. Tomcat writes to the Message Bus to cope with event or update storms.

The `db_connections` and `priority_db_connections` settings `$MOOGSOFT_HOME/config/servlets.conf` control the size of the database connection pool that Tomcat uses to connect to MySQL. You can increase either setting to potentially improve UI performance. Exercise caution when changing these values because increases to will typically increase CPU and memory usage of both the Tomcat and database processes. Too many connections can lead to an overload of transactions to the database which impacts other areas of the system.



JVM Performance

Integrations and LAMs, Moogfarmd, and Tomcat are all Java processes so you can tune the memory allocation pool settings for the JVM to optimize performance. This `-Xmx` setting defines the maximum allowed Java heap size of the process. The default memory allocation for a Java process is one quarter of the server's RAM.

For LAMs, integrations and Moogfarmd, you can add the `-Xmx` argument to the line in `$MOOGSOFT_HOME/bin/<lam name>` or `$MOOGSOFT_HOME/bin/moogfarmd` where the JVM is launched.

For example, to set the maximum Java heap size for the Moogfarmd process to 16 GB, add `"-Xmx16g"` to the `java_vm` command line in `$MOOGSOFT_HOME/bin/moogfarmd` as follows:

```
#Run app
$java_vm -server -Xmx16g -DprocName=$proc_name -
D$MOOGSOFT_HOME=$MOOGSOFT_HOME -classpath $java_classpath $java_main_class
"$@" &
```

For Tomcat, the default setting is 2GB. If you need to change it you can edit the service script `/etc/init.d/apache-tomcat`.

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: Incorrect Cisco Crosswork Situation Manager version

If an incorrect or outdated version is offered when installing Cisco Crosswork Situation Manager your Yum cache may need cleaning.

Run the following command and then re-attempt the installation:

```
yum clean all
```

Yum: HTTP Error 401 - unauthorized



If an attempt to install Cisco Crosswork Situation Manager fails with an error such as the following, check your username and password credentials are correct in the configured Cisco Yum repository.

```
https://<username>:<password>@speedy.moogsoft.com/repo/aiops/latest/repodata/repomd.xml: [Errno 14] HTTP Error 401 - Unauthorized
```

Yum: Problem making SSL connection

If an attempt to install Cisco Crosswork Situation Manager fails with an error such as the following:

```
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
```

You may need to update the NSS packages on your server. Run the following command and then re-attempt the installation.

```
yum -y update nss
```

Yum: MySQL conflict

If an attempt to install Cisco Crosswork Situation Manager fails with an error such as the following, it may be caused by a conflict with the MySQL libraries on the host.

```
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
-----
```

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 on which some MySQL components are already installed. The issue is often seen when trying to install moogsoft-db on a system with an existing MySQL installation.

Nginx: Installation problems

Error: Package: moogsoft-ui-7.2.0-123.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 0:7.2.0-123 will be an update
--> Processing Dependency: nginx >= 1.14.0 for package: moogsoft-ui-7.2.0-123.x86_64
--> Finished Dependency Resolution
Error: Package: moogsoft-ui-7.2.0-123.x86_64 (moogsoft-aiops)
Requires: nginx >= 1.14.0
```

Try using `--skip-broken` to work around the problem, or try:

```
rpm -Va --nofiles --nodigest
```

Alternatively, you could manually install the Nginx repo with the following command and then re-attempt the Cisco Crosswork Situation Manager installation.

```
rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-2.el7ngx.noarch.rpm
```

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
```

This returns the status disabled if the firewall is disabled.

2. To disable an active firewall:

```
setenforce 0
```

3. To check whether a port is open:

```
firewall-cmd --zone=public --query-port=8443/tcp
firewall-cmd --zone=public --query-port=8080/tcp
```

4. To open a port:



```
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 Nginx configuration directory 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 HTTP/SSL implementation.
LAMs, for example: socketlamd	Link Access Modules used for data ingestion. Service names may differ.
trapdlamd	At least one instance of a LAM is required for data feed.
newreliclamd	



moogfarmd	Core Cisco Crosswork Situation Manager system application.
mysqld	Database containing Cisco Crosswork Situation Manager data (database schemas etc.)
rabbitmq-server	Message system for Cisco Crosswork Situation Manager.
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 <service-name> stop
service <service-name> start
service <service-name> restart
```

Location of installation and log files

See [Configure Logging](#).

Generic Cisco Crosswork Situation Manager process not starting

No space left on the disk

- Check the file system with the 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`

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

Also see [Message System 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 ulimit settings for the RabbitMQ user by running the following command as root:

```
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              (-x) unlimited
```

- The above example shows ulimit settings that are likely too low for RabbitMQ
- As per instructions [here](#) it may be appropriate to increase the ulimit settings for "open files" and "max user processes" to at least **4096** for development/QA environments and **65536** for production environments

RabbitMQ: Unable to create connection



- "Unable to create connection" message appearing in a LAM, Moogfarmd, or Apache Tomcat logs
- This indicates that the process unable to connect to the Message Bus zone in RabbitMQ
- Check that the RabbitMQ server is running:

```
service rabbitmq-server status
```
- If the service is not running start it:

```
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:

```
rabbitmqctl list_vhosts
```
- Check the MooMS section 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 [Message System Deployment](#)).
- Restart the affected process.

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 initd files contain this setting):

```
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` indicates a syntax error in the LAM configuration file.



- Check the config file for syntax mistakes.

Look for:

- Missing commas
- Unbalanced quotes
- Compare the configuration file to a default configuration 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" error in /var/log/moogsoft/<lamd_name>.log 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 configuration file and restart the LAM.

Unresolvable hostname error

- "Host [hostname] unresolvable" error in /var/log/moogsoft/<lamd_name>.log means that the LAM is unable to resolve the host, or the hostname is incorrectly set.
- In the LAM config file, check the address property and correct any errors.
- Check that the /etc/hosts file contains an entry for the specified hostname.

Failed to open file error

- "Failed to open file [<path to file>] error in /var/log/moogsoft/<lamd_name>.log means that 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 configuration 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.



- You can do this with a JavaScript editor. Check the code to locate any syntax errors.
- Or if you have installed Node.js you can run the following command to locate the incorrect line in the code:

```
node $MOOGSOFT_HOME/bots/lambots/<path_to_filter_file>.js
```

- Edit `$MOOGSOFT_HOME/bots/lambots/<path_to_filter_file>.js` to resolve the error and restart the LAM.

Empty columns in alert lists

- Empty columns in alert lists may indicate incorrect field mapping assignments.
- Check field mappings at the bottom of the LAM configuration file.
- Edit the configuration file to properly map the field to the column name and then restart the LAM.

Socket LAM not processing

Processing mode set incorrectly

- The LAM may be set to Server mode rather than Client mode in the configuration file. For a description of mode types see [Socket LAM](#).
- Set the mode correctly in the configuration 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 debug mode and look for nested JSON.
- 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]" error in `/var/log/moogsoft/<lamd_name>.log` shows that the Logfile LAM cannot locate the log file to read.
- In the LAM configuration file check the target setting 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`



- "No file path specified" error in /var/log/moogsoft/<lamd_name>.log indicates a missing SSL path in the LAM configuration file.
- Check the value of the use_ssl property has been set correctly.
- If using SSL, check the following properties are correctly set:

path_to_ssl_files

ssl_key_filename

ssl_cert_filename

Nginx fails on startup if IPv6 not configured

Comment out the following references in two configuration files:

1. Go to /etc/nginx/conf.d
2. Edit out IPv6 references with a hash #:

Configuration 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:

service nginx status

- Check that Nginx is listening on port 443. Example expected output:

```
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:



```
service apache-tomcat status
```

Communication problem between the UI and MySQL database

- Check the MySQL service is running:

```
service mysqld status
```

- If MySQL is running on a different server, check that it is accessible from the Cisco Crosswork Situation Manager web server and the required permissions have been applied.

Authentication problem between the UI and MySQL database

- Check that the user exists in the MySQL moogdb.users table.
- Check that the username and password used for authentication are correct.

Search/Elasticsearch

See [Configure Search and Indexing](#) for more information.

ElasticSearch not running or generating errors (such as MySQL connection problems)

- Check that the Elasticsearch service is running:

```
service elasticsearch status
```

- Any errors are 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.

- List the configured cron jobs:

```
crontab -l
```

- Errors are written to /var/log/cron
- Depending on the intervals at which Elasticsearch re-indexes against the Cisco Crosswork Situation Manager database, it is possible that new alerts, Situations, threads or comments have not yet been indexed, and so will not be searchable.
- To change the interval manually:



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.SettingCommand.execute(SettingCommand.java:54)
[elasticsearch-5.6.9.jar:5.6.9]
    at
org.elasticsearch.cli.Command.mainWithoutErrorHandling(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 noexec directive in the /tmp mount. The solution is to remove the noexec directive, if it is practical to do so:

```
sudo mount /tmp -o remount,exec
```

Or set the following in /etc/sysconfig/elasticsearch:

```
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 that the product license has been applied.

RabbitMQ not running

- Check that the RabbitMQ server is running:

```
service rabbitmq-server status
```

Alert Builder not started

- Check that the "run on startup" setting for Alert Builder in the Moogfarmd configuration file is set to true.
- Check the Moogfarmd log /var/log/moogsoft/moogfarmd.log

Alert Builder misconfiguration

- Check the Alert Builder Moolet 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 performing any post-event processing that may be filtering out the events in the associated LAMbot.

No Situations Created

License not applied

- Ensure that the product license has been applied.

Sigaliser Moolet not running



- Check the Moogfarmd configuration file to ensure that the "run on startup" property for the Sigaliser used is set to true.

Incorrectly set "process output of" property

- In the Moogfarmd configuration file, check that the "process output of" setting for the Sigaliser used lists the correct Moolet.
- Check that the Moolet listed in the "process output of" property is running.

Sigaliser Moolet configuration is too restrictive or too open

If Moolet settings are too restrictive or too open they may not produce Situations. See [Sigaliser Moolet](#) 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	Sent to the browser when a request has been processed successfully.
202	Accepted	Sent to the browser when the request was well constructed but the server was unable to process it.
400	Bad Request	Sent to the browser when request parameters are invalid or missing.
401	Unauthorized	Returned when the user is not authenticated.
403	Forbidden	Returned when the user is forbidden to perform a specific action.
404	Not Found	Sent to browser when a requested servlet path could not be found.
409	Conflict	A conflict has occurred that can potentially be resolved by the user. For example an incorrect template name.
500	Internal Server Error	Sent to the browser when there is an unexpected exception or problem on the server side.
503	Service Unavailable	Used when the server is overloaded. This is 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. A single 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 defined:

Code	Description	HTTP Status
1000	General server error.	500
1001	Service unavailable.	503
1002	Database unavailable.	500
1003	Service busy.	202

Validation Error Status Codes

The following validation error status codes are 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 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 defined:

Code	Description	HTTP Status
4000	General conflict error.	409
4001	Duplicate name. For example duplicate template name, invite already sent.	409

Content Error Status Codes

The following content error status codes are 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 the Cisco Crosswork Situation Manager version).	404
5005	UI being ahead of the UI (in terms of the Cisco Crosswork Situation Manager version).	40

JSON Response Body (Non-HTTP 200)

The response body will contain the detailed application error status information. The additional information is optional for some status codes.

The format of the JSON is as follows:

```
{
  "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 <cause>". For example "Create situation failed due to invalid parameters".

JSON Addition Information

The format of the JSON additional information depends on 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 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>  
}
```

Ngnix Error Messages

97: Address family not supported by protocol

This an IPv6 error. Either configure IPv6 or comment it out.

To comment it out, go to /etc/nginx.conf.d and comment out the IPv6 references with a hash #:



Configuration File	Section
--------------------	---------

moog-default.conf	<code>listen 80 default_server;</code> <code>#listen [::]:80 default_server;</code>
moog-ssl.conf	<code>listen 443 ssl default_server;</code> <code>#listen [::]:443 ssl;</code>

502 Bad Gateway (nginx/1.14.0)

Fails to load resource and server responds with a status of 502 (Bad Gateway).

Check that you have used the correct components for the version you are installing.

```
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
```