Cisco Crosswork Situation Manager 7.3.x Clustering Algorithm Guide

Powered by Moogsoft AIOps 7.3

Clustering Algorithm Guide

Sigalisers are the clustering algorithms in Cisco Crosswork Situation Manager that group alerts based on factors such as time, language, similarity and proximity.

The clustering algorithms available include:

- Cookbook
- Tempus
- Feedback

You can configure and run multiple different clustering algorithms on the same instance of Cisco Crosswork Situation Manager. The algorithms you choose depend on your specific use cases and the type of Situations you want your operators to receive.

You can also apply entropy and Vertex Entropy calculations to add another degree of filtering to the alerts you want to correlate. For example, you can use an entropy threshold if you want to exclude alerts with low operational value or include alerts with high operational value. See Vertex Entropy and Entropy for more details.Entropy Overview

Cookbook

Cookbook is a clustering algorithm that creates clusters defined by the relationships between alerts and their attributes. See Cookbook for more information.

Type: Attribute-based clustering.

Use cases: You can use Cookbook if you want more control in how you correlate alerts based on patterns in the text similarity. Example use cases include:

- Grouping alerts with a similar description and from the same application or service.
- Grouping alerts from the same host or location.
- Topology-based correlation using Vertex Entropy.

Benefits: Cookbook offers the following advantages:

- Very customizable and configurable using Recipes.
- Able to create Situations when an alert exceeds a defined rate of occurrence.
- Can include and exclude alerts that meet specific criteria such as Vertex Entropy.
- Able to partition alerts into Situations using textual similarity-based comparison.
- Possible to base alert clustering on topological relationships.

Configuration: To configure Cookbook Recipes and Cookbook via the Cisco Crosswork Situation Manager UI, see Configure a Cookbook Recipe and Configure a Cookbook Graze API.

Tempus

Tempus is a time-based algorithm that clusters alerts into Situations based on the similarity of their timestamps. See Tempus for more information.

Type: Time-based clustering.

Use cases: You want to match alerts based on patterns in their timestamps or on a timeline. Use Tempus if you want your alerts to be clustered in real-time. The logic behind Tempus is that a triggering event causes additional subsequent failures within a short timeframe. Works well in scenarios where there is a causal chain such as:

- Cascading failures
- Performance failures
- Brownouts

Benefits: Tempus offers the following advantages:

- No enrichment required. See /document/preview/11750#UUIDa079e66cf5d05f6b29d33de8e0ea50a5.Enrichment
- Good for availability alerts.
- Good for performance alerts.

Configuration: To configure Tempus via the Cisco Crosswork Situation Manager UI, see /document/preview/112936#UUID16515e8e9d0e1c6d82f01cefaadc0374. You can also configure Tempus via the Graze API.Configure TempusGraze API

Feedback

Warning: Feedback is a Beta feature.

Feedback is the neural-based algorithm that learns and unlearns actions based on user feedback. See Feedback for more information.

Type: Neural/learns user feedback.

Use cases: Feedback is currently a prototype and should not be used in production environments. You can use it if the other clustering algorithms did not correlate anything, as you can teach it what to cluster. For example, if you have a set of alerts that you want to cluster but they didn't cluster through time, attribute similarity or topological proximity, you can teach the system and it learns to cluster those alerts.

Alternatively, you might want to use Feedback if you want to manually create Situations and teach Cisco Crosswork Situation Manager to cluster the same type of alerts. Another use case is to use Feedback alongside Tempus. If you have several team members looking at time-based correlation with an inherent degree of fuzziness, they can use Feedback to train the system to remember good Situations and forgot about bad Situations and persist that behavior in future. For example, you could teach it to remember when there was a server failure but to ignore the printer ink failure and persist that behavior.

Benefits: Feedback offers the following advantages:

- No enrichment required. See /document/preview/11750#UUIDa079e66cf5d05f6b29d33de8e0ea50a5.
- Allows operators to push domain knowledge back into the system.
- Can be trained to only create the Situations you are interested in.

Configuration: Both UI and backend configuration. See Configure Feedback for more information.

Cookbook

Cookbook is a deterministic clustering algorithm in Cisco Crosswork Situation Manager that creates Situations defined by the relationships between alerts.

You can configure Cookbook to cluster alerts into Situations if they have specific characteristics such as temporal or topological proximity. Cookbook filters can include characteristics such as the following:

- Class or type
- Description
- Server priority
- Geographical location
- Environment classification

Each Cookbook is a collection of Recipes: sets of configurable filters, triggers, and other calculations such as priority ordering and entropy threshold. A Cookbook can run multiple Recipes concurrently to process the incoming event stream and produce a variety of Situations. A Cisco Crosswork Situation Manager deployment may include multiple instances of Moogfarmd, each of which can run multiple Cookbooks.

Configure Cookbooks and Recipes

To configure Cookbooks and Recipes via the Cisco Crosswork Situation Manager UI, first configure your Recipes and then configure the Cookbooks that you want to use these Recipes in. See Configure a Cookbook Recipe Configure a Cookbook for details.

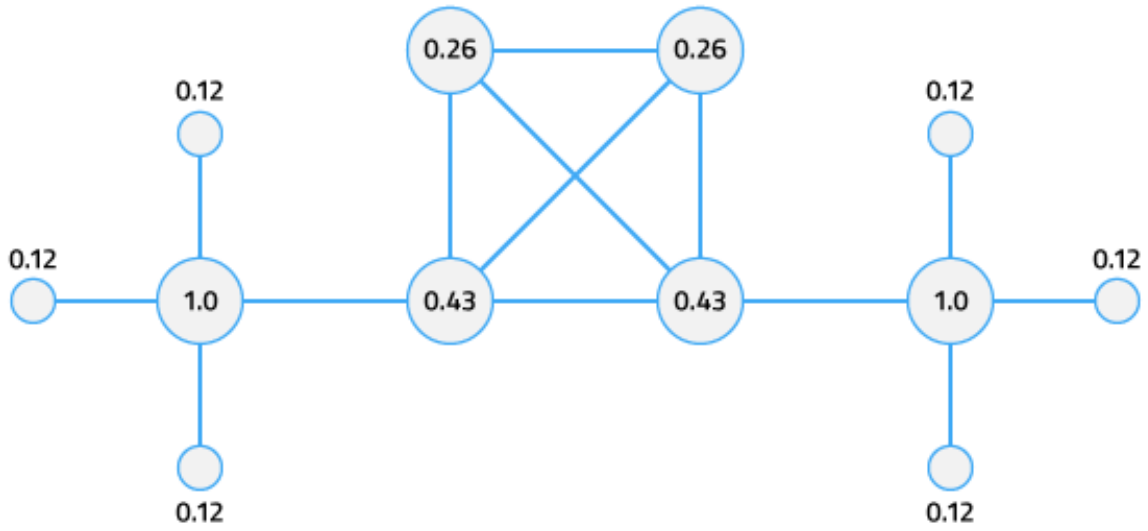
You can also configure a Cookbook and its Recipes via the Graze API.

Cookbooks configured in the UI and the Graze API can run concurrently.

Vertex Entropy

Vertex Entropy is a Cisco Crosswork Situation Manager algorithm that indicates the critical nodes within your network and their tendency to produce important events.

You can use Vertex Entropy if you want to cluster alerts into Situations based on their topological importance. Once the calculation is run against a topological map of the connected nodes in your network, it applies a Vertex Entropy value for each node or "vertex".



This diagram shows the Vertex Entropy values for a network of 12 connected nodes. A Vertex Entropy value of 1.0 indicates a node of highest topological importance.

Note: Node: A device or base unit that forms part of a larger network. This is known as a 'vertex' in graph theory.

Link: A connection between two directly connected nodes. This is known as an 'edge' in graph theory.

Hop: A jump between two directly connected nodes.

Set up Vertex Entropy

To set up Vertex Entropy within Cookbook Recipes, see /document/preview/114709#UUID1bb978536bbf4215b156771ae826d901Set Up Vertex Entropy

Tempus

Tempus is a time-based algorithm in Cisco Crosswork Situation Manager which clusters alerts into Situations based on the similarity of their timestamps.

The underlying premise of Tempus is that when things go wrong, they go wrong together. For example, if a core element of your network infrastructure such as a switch fails and disconnects then it affects a lot of other interconnected elements which send events at a similar time.

Tempus uses the Jaccard index to calculate the similarity of different alerts. It also uses community detection methods to identify which alerts with similar arrival patterns it should cluster into Situations.

As Tempus is time-based, you should not use it to detect events relating to the slow or gradual degradation of a service from disks filling up or CPU usage.

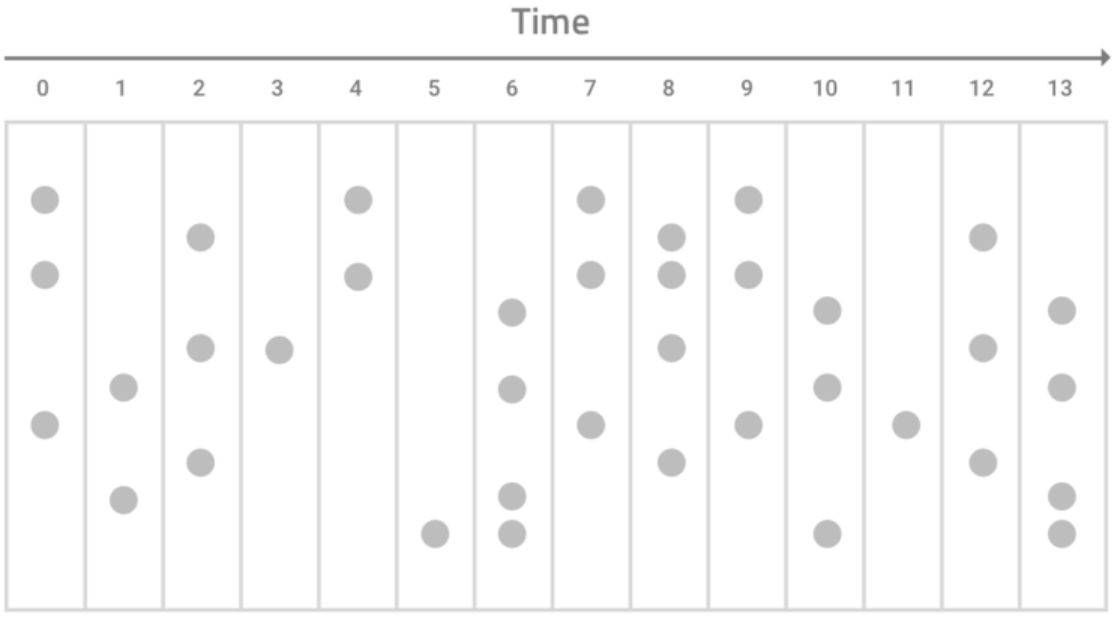
One advantage of Tempus is it only uses event timestamps for clustering so no alert enrichment is required.

Time-based clustering

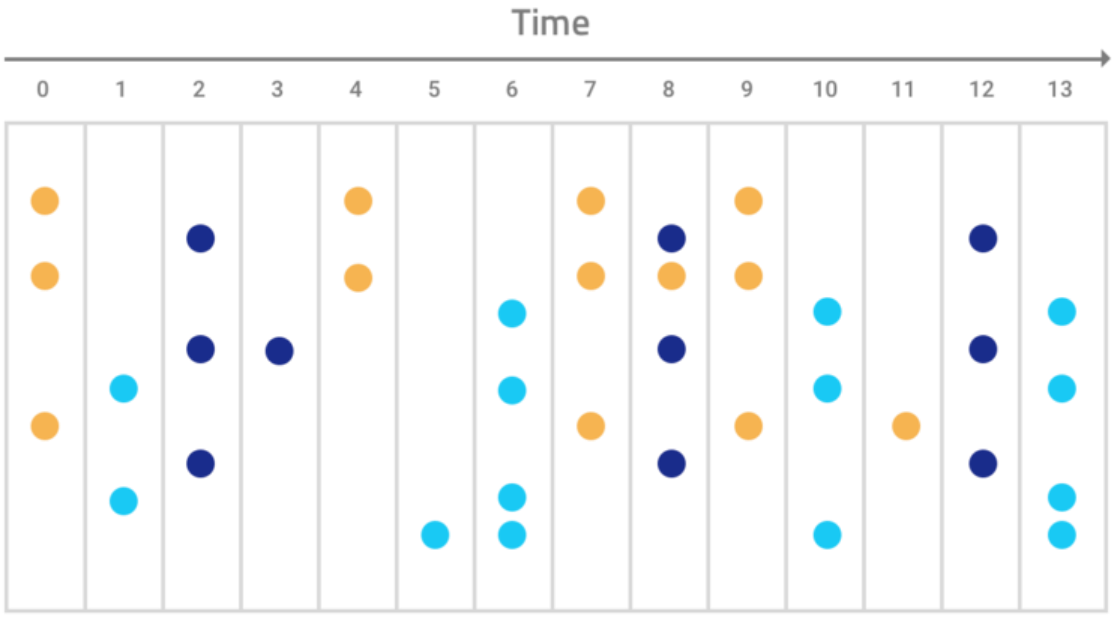
Cisco Crosswork Situation Manager applies Tempus incrementally to alerts as it ingests them so that it can create Situations in real-time.

The diagrams below show how Tempus sorts and then groups alerts with similar timestamps into Situations.

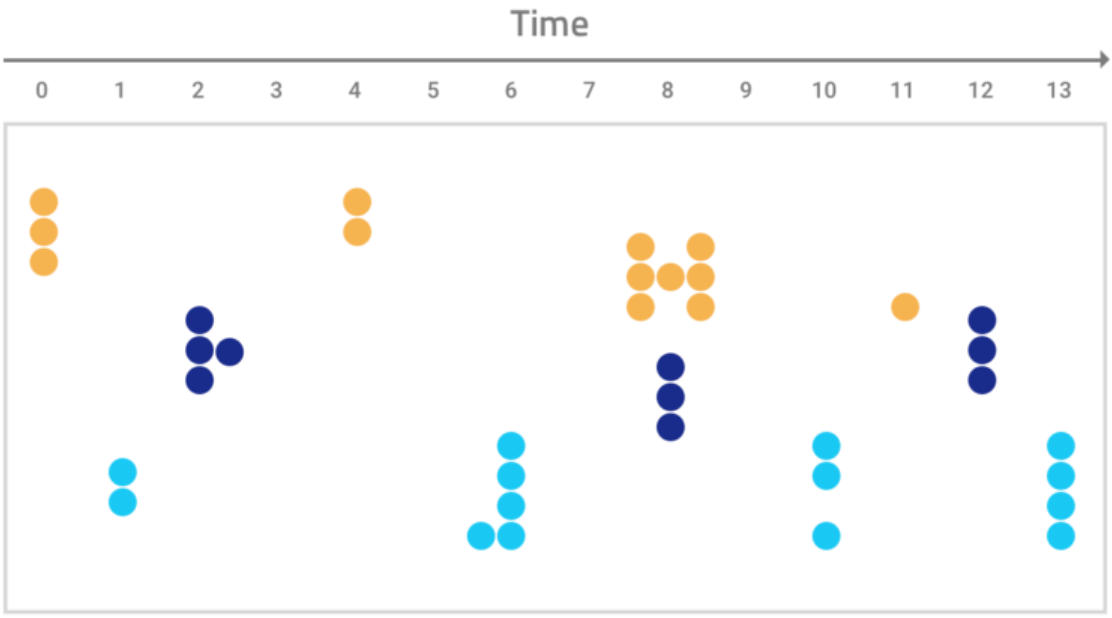
Raw alerts from the Moolet chain, for example, the Alert Builder or Alert Workflows, arrive over a period of time. These are shown as gray dots in the diagram below:



Tempus identifies and sorts which alerts have similar arrival patterns:



Tempus clusters alerts with similar arrival patterns into Situations:



Configure Tempus

To configure Tempus via the Cisco Crosswork Situation Manager UI, see /document/preview/112936#UUID16515e8e9d0e1c6d82f01cefaadc0374.Configure Tempus

You can also configure Tempus using the Graze API.

Feedback

Warning: Feedback is a prototype feature and is not recommended or supported for use in production environments.

Feedback is a clustering algorithm that creates Situations by learns from feedback that users add in the Cisco Crosswork Situation Manager UI.

You can configure Feedback by training it to identify the type of alerts you want it to cluster into Situations. When Feedback creates a Situation, it creates a neural network or "brain" that remembers the contents of the Situation. You can train each brain by identifying which alerts you want Feedback to include and exclude from Situations.

Feedback learns and unlearns by example, remembering what users identify as Situations and what it should not include in future Situations. For example, if Feedback creates a seed Situation with alerts for a host down error and a database restart error with the same IP address, it also creates a brain for that Situation. If Feedback clusters three new alerts based on this brain and you want to delete one of these alerts from the Situation, this retrains the brain.

The main risk with Feedback is if you provide it with erroneous feedback, the Situations it produces will also be erroneous.

To configure Feedback, see Configure Feedback.

Configure Feedback

Warning: Feedback is a prototype feature and is not recommended or supported for use in production environments.

Feedback is a supervised machine learning algorithm that creates Situations based on user feedback.

You can enable and configure Feedback in the **$MOOGSOFT_HOME/config/moolets/feedback.conf** configuration file. After you enable Feedback, you can select and edit the Feedback brain via the System Settings in the Cisco Crosswork Situation Manager UI.

Refer to Feedback Reference to see all available properties.

Before You Begin

Before you set up Feedback, ensure you have met the following requirements:

- Your LAMs or integrations are running and Cisco Crosswork Situation Manager is receiving events.
- You have configured the Moolet that is the source of the alerts for Feedback. You select the source using the **process_output_of** property.

Configure Feedback

Edit the configuration file at **$MOOGSOFT_HOME/config/moolets/feedback.conf.**

See Feedback Reference for a full description of all properties. Some properties in the file are commented out by default.

1. Provide a name for algorithm:
   - name:Name of the Feedback Moolet.
2. Configure Feedback's behavior for when it starts and stops running:
   - run_on_startup: Determines whether Feedback runs when Cisco Crosswork Situation Manager starts.
   - process_output_of: Defines the Moolet sources of the alerts that Feedback processes.
3. Configure the Feedback algorithm and how it clusters alerts:
   - membership_limit:Maximum number of Situations that an alert can be a member of.
   - scale_by_severity: Feedback treats alerts with a high severity like alerts with a high entropy value.
   - entropy_threshold: Minimum entropy value an alert must have in order for Feedback to include it in a Situation.
   - single_matching: Match alerts to the most suitable neural network.
4. Configure the neural networks:
   - inputs:Alert attributes that you are interested in matching and want Feedback to consider when learning a Situation.
   - learn_queues: Collection of actions you want to trigger learning.
   - unlearn_queues: Collection of actions that can trigger the removal of a neural network.
   - rating_threshold:Measures when to trigger learning or unlearning.
   - match_strategy: Determines how Feedback matches alerts.
   - precision: Determines how precisely you want to train the brain.
   - tolerance: Determines the degree of error an alert can have to create a new Situation.
   - window: Determines the length of time in seconds that Feedback analyzes alerts and develops a Situation each time it runs.

Restart the Moogfarmd service to activate any changes you make to the configuration file. See Control Moogsoft AIOps Processes for further details.Control Processes

Example

The following example demonstrates a simple Feedback Sigaliser:

```
{
            # Moolet
            name                : "Feedback",
            classname           : "com.moogsoft.farmd.moolet.feedback.CFeedback",
            run_on_startup      : false,
            process_output_of   : "MaintenanceWindowManager",
            membership_limit    : 1,
            scale_by_severity   : false,
            entropy_threshold   : 0.0,
            single_matching     : false,
            inputs              : [ "source","description" ],
            learn_queues        : [ "manual_create","rated" ],
            unlearn_queues      : [ "rated" ],
            rating_threshold    : 3,
            precision           : 92.0,
            tolerance           : 0.01,
            exact_match         : false,
            window              : 60
   }
```

Feedback Reference

Warning:

> Feedback is a prototype feature and is not recommended or supported for use in production environments.

This is a reference for the Feedback clustering algorithm. The Feedback configuration properties are found in **$MOOGSOFT_HOME/config/moolets/feedback.conf.**

Cisco recommends you do not change any properties that are not in this reference guide.

Moolet

name

Name of the Feedback Sigaliser algorithm. Do not change.

Type: StringRequired: Yes

Default: **"Feedback"**

class

Moolet class name. Do not change.

Type:StringRequired: Yes

Default:**"com.moogsoft.farmd.moolet.feedback.CFeedback"**

run_on_startup

Determines whether Feedback runs when Cisco Crosswork Situation Manager starts. If you enable this property, Feedback captures all alerts from the moment the system starts, without you having to configure or start it manually.

Type: Boolean

Required: No

Default: **`false`**

process_output_of

Defines the Moolet source of the alerts for Feedback.

Type: List

Required: Yes

One of: **`AlertBuilder, AlertRulesEngine, MaintenanceWindowManager, EmptyMoolet`** Default: "**`MaintenanceWindowManager`**"

Algorithm

membership_limit

Maximum number of Situations an alert can be part of. This does not impact alerts in merged Situations. Smaller limits result in fewer Situations with many alerts and many Situations with fewer associated alerts. Larger limits result in many Situations with few alerts and a few Situations with many alerts. The optimal value is between 1 and 5.

Type: Integer

Required: Yes

Default: 1

scale_by_severity

Feedback treats alerts with a high severity like alerts with a high entropy value. Cisco Crosswork Situation Manager divides the severity number by the maximum severity (5) to calculate the scale. For example, for an alert with minor severity, the entropy is 3/5.

Type: Boolean

Required: No

Default: False

entropy_threshold

Minimum entropy value that an alert must have for Feedback to consider it for clustering into a Situation. Feedback does not include any alerts with an entropy value below the threshold in Situations. Set to a value between 0.0 and 1.0. The default of 0.0 means Feedback processes all alerts.

Type: Decimal

Required: No

Default: 0.0

single_matching

Enable **`single_matching`** for Feedback to match alerts to the most suitable neural network.

Type: Boolean

Required: No

Default: **`false`**

Neural Network (Brain)

inputs

Alert attributes that you are interested in matching and want Feedback to consider when learning from Situations.

Type: Array

Required: Yes

Default: **`[ "source","description","severity","manager" ],`**

learn_queues

Collection of actions that you want to trigger learning. See User Actions for available actions. For example, you might want Feedback to learn when a user creates a Situation or gives a Situation a high rating.

Type: Array

Required: Yes

Default: **`[ "manual_create","rated","merge_create","split_create","annotated","diagnosed","refined"],`**

unlearn_queues

Collection of actions that can trigger the removal of a neural network. For example, you might want Feedback to unlearn when a user gives a Situation a low rating.

Type: Array

Required: Yes

Default: **`[ "rated","split" ],`**

rating_threshold

Measures when to trigger learning or unlearning determined by the star rating out of five that users give the Situations in the Cisco Crosswork Situation Manager UI. By default, a rating of three or higher triggers learning. A rating of lower than three triggers unlearning.

Type: Array

Required: Yes

Default: **`3`**

precision

Determines how precisely you want to train the brain. Increase this value if Feedback is not producing accurate results. Cisco does not recommend reducing this below 92%.

Type: Integer (%)

Required: Yes

Default: **`92.0`**

tolerance

Determines the percentage degree of error an alert can have to create a new Situation. For example, if the neural network has a trained value of 0.95 and the new alert has a value of 0.8, the tolerance needs to be 0.15 or lower for Feedback to create a new Situation.

Type: Integer (%)

Required: Yes

Default: **`0.01`**

window

Determines the length of time in seconds that Feedback analyzes alerts and a Situation develops each time it runs. Defaults to 60 seconds.

Type: Integer

Required: Yes

Default: **`60`**

User Actions

Feedback can learn or unlearn from the following user actions:

- **`manual_create`**: User has manually brought alerts together to create a Situation.
- **`merge_create`**: User has merged two or more Situations together. Feedback learns the newly created Situation.
- **`split_create`**: User has created some new Situations by splitting an existing Situation. Feedback learns from the newly created Situations.
- **`annotated`**: User has started a discussion thread or added a comment on the Situation or has set a Situation's description.
- **`diagnosed`**: Tool has been run on the Situation.
- **`closed`**: Situation has been closed or resolved.
- **`refined`**: Alerts have been added to or removed from a Situation.

Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see *What's New in Cisco Product Documentation*.

To receive new and revised Cisco technical content directly to your desktop, you can subscribe to the What's New in Cisco Product Documentation RSS feed. The RSS feeds are a free service.

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco Trademark

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: www.cisco.com/go/trademarks. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)

Cisco Copyright