White Paper

Cisco IOS Software Embedded Event Manager Harnesses Network Intelligence to Increase Availability

Last updated: December 2005

The Cisco IOS[®] Software Infrastructure provides a method of triggering pre-programmed local actions upon detection of specific events. Cisco IOS Software Embedded Event Manager harnesses network intelligence through event detectors and takes action according to pre-defined policies, resulting in increased manageability, control, and resiliency.

Organizations are increasingly reliant on advanced information systems to build products, serve customers, and achieve goals. IT departments are under pressure to ensure continuous access and Cisco customers frequently cite the availability of critical systems as a high priority. Because today's systems, data stores, and user communities are often highly distributed, combining effective network management with a resilient network increases the availability of critical systems. A combination of redundancy, automatic traffic rerouting, and effective network management systems reduces the impact of faults and improves both actual and perceived service levels.

When customers need uninterrupted service, the network must circumvent failures without manual intervention. Operators, who will always be necessary, need tools that provide automated recoveries and that aid in network management. In order to achieve the required levels of dependability and resiliency, it is critical to increase the level of automation available locally within the network device, improving fault detection and recovery capabilities.

Fault detection and recovery must be immediate and efficient. Rapid identification and accurate analysis of fault conditions is imperative before corrective action can begin. Fortunately, with advanced planning and design, redundant paths and rerouting of traffic compensate for many problems and anomalies. Yet the fault recovery process can still be improved by customizing local actions based on the network's current state, and the sequence of events leading up to the problem. The network can be made more dependable by integrating diagnostic routines and triggers, along with programmable actions that contain and alleviate faults.

Network management has been an important part of network operations since the first time-sharing and distributed systems became prevalent. Many network operations centers rely on a centralized, hierarchical management model. Typically, network management servers at some central location are notified of changes in network status via Simple Network Management Protocol (SNMP). The system then attempts to sort through the data it received in order to ascertain the root cause of the problem. Some predefined action might be taken once the cause is identified. This may involve notifying the network operations staff, creating a trouble ticket, or even communicating with remote network equipment via scripted actions that use SNMP or telnet.

Note that there are some problems associated with this model:

- Can be difficult to determine the root cause of the problem
- Volume of data and notifications can be daunting
- Conflicting or compounded information makes it hard to determine the original cause, even with automation tools
- Systems and personnel can become confused, particularly if more than one problem occurs simultaneously

Cisco IOS Embedded Event Manager (EEM) is designed to help resolve these issues by distributing automation and delegating authority to the point closest to the problem. By harnessing the extensive network knowledge within Cisco IOS Software running on the routers and switches, appropriate actions can be invoked more swiftly and efficiently than is possible with a purely centralized network management model.

This document is targeted to information technology professionals, network designers, network managers, and network operations and management staff responsible for the availability of critical systems. It describes Cisco IOS Software EEM and the associated policy engines.

INTELLIGENCE EMBEDDED WITHIN CISCO IOS SOFTWARE

The renewed Cisco IOS Software Infrastructure contains an EEM component and methods to invoke custom, local actions trigged by defined events. There is also a programmable scripting language based on Tool Command Language (Tcl). EEM provides a window to the detailed network knowledge embedded within Cisco IOS Software. Collectively, EEM and its associated components allow network operators to harness the vast network operational data and hardware and software diagnostics embedded within Cisco IOS Software. It represents a framework for monitoring and detecting certain conditions that might impact network services. And it includes methods to program specific actions to take upon detection of certain events. Because it is integrated within Cisco IOS Software, the Cisco IOS Software EEM has intrinsic knowledge of the state of the network from the point of view of the device on which it is operating. This intelligence, combined with the programmability of EEM, provides a powerful facility that can be leveraged in many ways.

The local scripting capability can be applied to many scenarios that previously required programming and scripting at a central network management station. Eliminating the dependence on a remote network management system makes new fault handling capabilities possible, even when the network connection to the management system is impaired.

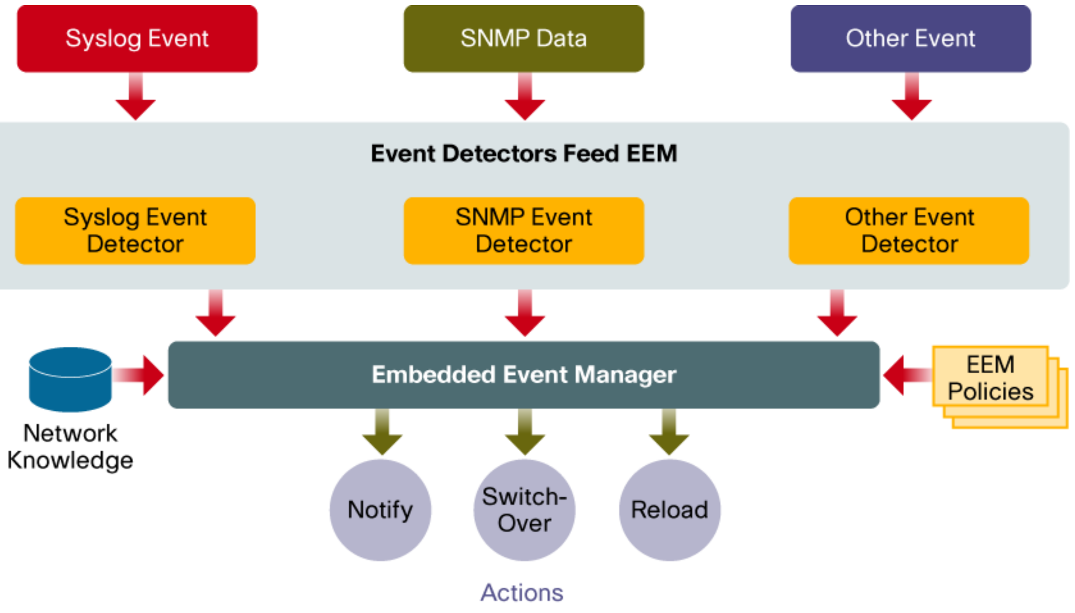
Cisco IOS Software EEM creates a new environment for distributed network management. It is anticipated that systems integrators, managed service providers, and other network and systems management vendors will embrace this capability and exploit the programmable interface to enhance their product or service. The integration of Cisco IOS Software EEM with network operations procedures will result in improved manageability of network resources, ultimately improving the resilience of network services and the availability of critical systems.

Cisco IOS Software EEM is available now for many Cisco routing and switching products. It is part of Cisco IOS Software and available in Cisco IOS Software Releases 12.0S, 12.2S, 12.2SX, 12.3T, 12.4 and 12.4T. The function initially available in Releases 12.0(26)S and 12.3(2)T is referred to as EEM Version 1.0. EEM Versions 2.0, 2.1, and 2.2 are available for certain releases with more being added. EEM continues to be enhanced and the added function will be made available throughout the product line over time. The function is available or planned for most hardware products that run Cisco IOS Software. As always, consult Cisco Feature Navigator for specific release availability and see ITD Product Management for roadmap details.

CISCO IOS SOFTWARE EEM ARCHITECTURE

Figure 1 illustrates the major components of the Cisco IOS Software EEM subsystem. The EEM acts as a clearinghouse for all local events and provides a mechanism to trigger Cisco or user-written scripts to take appropriate action. EEM is integrated within Cisco IOS Software as an independent subsystem. Events can be related to some type of fault condition or can simply be some occurrence or state change that is deemed interesting. Event detectors are Cisco IOS Software components that detect specific network-related changes and determine the nature of the event. The event detectors inform the event manager, with a prescribed internal communication mechanism.

Figure 1. Cisco IOS Embedded Event Manager Architecture



Event Detectors

Cisco IOS Software EEM event detectors provide an interface between the monitored agent and the action policies. Event detectors determine that a particular event has occurred and notify the event manager. Several event detectors have been created and others will be added in the future. At the time this document was written (November 2005) there were fifteen event detectors (see Table 1).

Table 1. Partial List of Event Detectors

Event Detector Name	Description
Cisco IOS Command Line Interface (CLI) Event Detector	Triggers policies based on commands entered via the CLI. Uses a regular expression match.
Cisco IOS Counter Event Detector	Policies can be triggered based on a change of the designated counter. The counter event detector is used to manipulate counters named by the policy writer and internal to EEM.
Cisco IOS Redundancy Facility (RF) Event Detector	The Cisco IOS RF provides for detection of hardware and software failures related to the stateful switchover service. This event detector will trigger policies based on the RF state change. It is also used to initiate switchovers as a result of a policy action.
Cisco IOS Resource Threshold Event Detector	Triggers policies based on global platform values and thresholds. Includes resources such as CPU utilization and remaining buffer capacity.

Event Detector Name	Description
Cisco IOS Timer Services Event Detector	Policies can be scheduled to occur at the designated time or interval. This event detector provides an option to create time-based triggers similar to the UNIX CRON facility.
Cisco IOS Watchdog/System Monitor Event Detector	Triggers policies based on certain conditions relative to a certain Cisco IOS Software process or subsystem's activity.
EEM Application Specific Event Detector	Application specific events can be detected or set by a Cisco IOS Software subsystem or a policy script. This provides the ability for one policy to trigger another policy.
Interface Counter Event Detector	Policies can be triggered based on the specific interface counter. Includes thresholds.
Online Insertion and Removal (OIR) Event Detector	Triggers policies based on hardware installation and removal activity.
Routing Event Detector	Triggers policies based on routing protocol events.
Simple Network Management Protocol (SNMP) Event Detector*	Triggers policies based on the associated SNMP MIB variable. Includes MIB variable thresholds.
Syslog Event Detector*	Triggers policies based on the regular expression match of a local Syslog message.
System Manager Event Detector	Triggers policies based on conditions relative to a certain Cisco IOS Software process or subsystem's activity. This event detector is unique to Cisco IOS Software Modularity for the Cisco Catalyst 6500 Series Switch.

* The SNMP and Syslog event detectors are available in EEM version 1.0. The rest of the event detectors are included in later releases.

Policy Engines

Policy engines are the methods of programming available with EEM. There are two policy engines defined. One is the Cisco IOS Software CLI applet interface and the other is the Tcl subsystem and interpreter. A policy consists of an event trigger coupled with some defined action. A policy must be registered with one of these two policy engine facilities. Once registered, the event manager invokes the policy after corresponding event detector detects the trigger event. Policies reference environment variables to determine the specifics of particular events when the policy gets control.

Cisco IOS Software EEM Versions

Cisco is delivering Cisco IOS Software EEM in phases: EEM version 1 and EEM version 2. Cisco IOS Software EEMv2 builds on the initial capabilities and offers expanded functionality.

Cisco IOS Software EEMv1.0

Cisco IOS Software EEMv1.0 incorporates two event detectors: SNMP events and Syslog events. These two event detectors can handle a significant number of events. The extensive Management Information Base (MIB) variables and counters are available as action triggers. An EEM policy can be defined to take action upon match on the SNMP MIB value or when the MIB object crosses a specified threshold. Likewise, the Syslog event detector opens the door to trigger actions based on regular expression pattern match against any Cisco IOS Software Syslog message.

EEMv1 allows policies to be defined using the Cisco IOS Software CLI applet. The following policy actions can be established:

- Generate prioritized Syslog messages
- Generate a Cisco Network Services (CNS) event for upstream processing by Cisco CNS devices
- Reload the Cisco IOS Software
- Switch to a secondary processor in a fully redundant hardware configuration

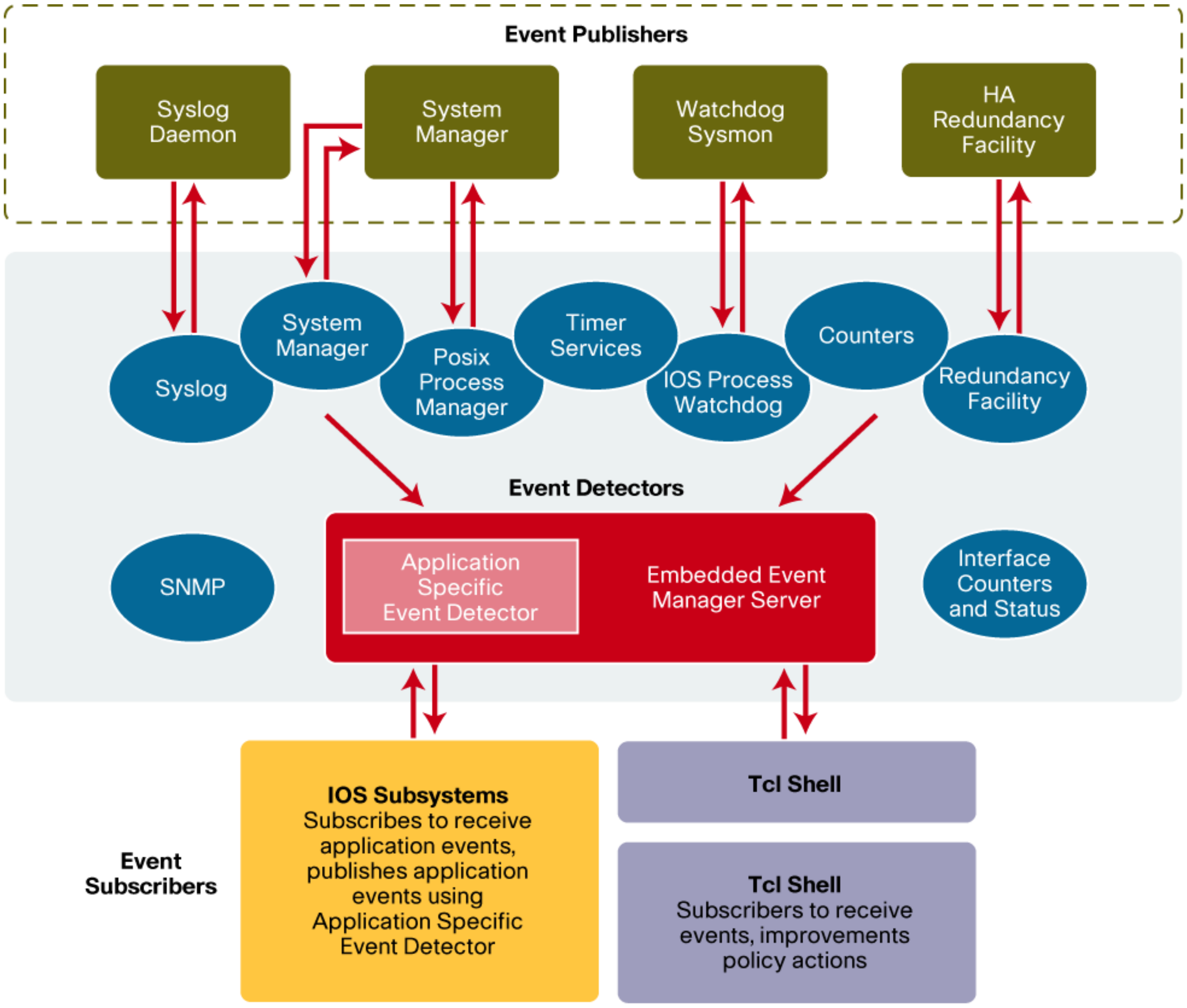
Cisco IOS Software EEMv2

Cisco IOS Software EEMv2 (Figure 2) adds flexibility, event detectors, and programmable actions with the Tcl subsystem within Cisco IOS Software. Version 2 functionality is now available in several releases and for Cisco routing and switching products such as the Cisco Catalyst 6500 Series Switches and the integrated services routers. Over the next several releases, Cisco IOS Software EEM functionality will converge to a common set of capabilities applicable to both modular, microkernel-based Cisco IOS Software, and the cooperative multitasking Cisco IOS Software versions.

For additional information about Cisco IOS Software Modularity for the Cisco Catalyst 6500 Series Switch, please visit: http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/product_promotion0900aecd80312844.html

Figure 2 also depicts two new concepts: event publishers and event subscribers. This reflects the more extensible architecture of Cisco IOS Software EEM v2. In a broad sense, Cisco IOS Software EEM interacts with other Cisco IOS Software routines (event detectors) that actively monitor the system for events and fault conditions. When an event occurs that matches a previously registered event specification, it is “published”, and actions defined within a policy are executed. Other Cisco IOS Software routines, modules, or “features” may also interact with the Cisco IOS Software EEM subsystem using internal APIs. Routines that perform this are Tcl-based policies or other Cisco IOS Software modules that have been so programmed. Generically, these routines are categorized as event subscribers.

Figure 2. EEM Version 2 Event Publishers and Subscribers


Event Publishers
Syslog Daemon
System Manager
Watchdog Sysmon
HA Redundancy Facility
Syslog
System Manager
Posix Process Manager
Timer Services
IOS Process Watchdog
Counters
Redundancy Facility
Event Detectors
SNMP
Application Specific Event Detector
Embedded Event Manager Server
Interface Counters and Status
Event Subscribers
IOS Subsystems
Subscribes to receive application events, publishes application events using Application Specific Event Detector
Tcl Shell
Tcl Shell
Subscribers to receive events, improvements policy actions

The remaining event detectors listed in Table 1 are included in EEMv2. Other event detectors are in development or planned and will be available in future Cisco IOS Software releases. Cisco EEMv2 allows the policy actions to be defined with the Cisco IOS Software CLI applet listed for Cisco IOS Software EEMv1 or policies can be programmed using Tcl.

CREATING POLICIES USING APPLETS

An applet is a simple policy that is defined within the CLI configuration. Defining policies using the applet interface does not require any programming effort or experience. It can be used to define simple policies that are triggered by specific events. Cisco IOS Software EEMv1.0 supports only the applet interface for policy definition.

The following steps illustrate how easy it is to define a policy using the applet interface.

Step 1. Register policy

The first step in creating a policy action is to register the policy with the applet policy engine. This is done using the command:

```
event manager applet <applet-name>
```

Step 2. Establish event trigger

Define the event that triggers the policy using the following command under the event manager applet sub-mode:

```
event snmp
```

or:

```
event syslog
```

Step 3. Define action

Define the action to take upon detection of the event as established in Step 2. This is accomplished using one of the action commands:

```
        action cns-event
        action force-switchover
        action reload
        action syslog
```

Please see the appropriate Cisco documentation for detailed information on the commands listed above.

Simple Applet Policy Example

The following example will illustrate how an applet policy can be defined. Suppose the user needs to define a custom Syslog message triggered whenever a particular Cisco IOS Software Syslog message is issued. The message should only be triggered from the following configuration Syslog message:

```
%SYS-5-CONFIG_I: Configured from console by console
```

The configuration command below will register an applet policy called **“eemCfgEvLog”**, define the event trigger as a Syslog message matching a regular expression, and define an action to issue a custom Syslog message.

```
router1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
router1(config)#event manager applet eemCfgEvLog
router1(config-applet)#event syslog pattern “\%SYS-5-CONFIG_I: Configured”
router1(config-applet)# action A01 syslog priority notifications msg “Configuration event detected”
```

The applet policy can be displayed using the following command:

```
router1#show event manager policy registered
No.  Type    Event Type          Time Registered           Name
1    applet  syslog              Tue Oct28  20:33:10 2003  eemCfgEvLog
 pattern {\%SYS-5-CONFIG_I: Configured}
 action A01 syslog priority notifications msg Configuration event detected
router1#
```

The effects can be seen by going in and out of configuration mode and displaying the log:

```
router1#conf t
Enter configuration commands, one per line. End with CNTL/Z.
router1(config)#^Z
router1# sh log
Syslog logging: enabled (10 messages dropped, 13 messages rate-limited,
                0 flushes, 0 overruns, xml disabled, filtering disabled)
    Console logging: level debugging, 76 messages logged, xml disabled,
                     filtering disabled
    Monitor logging: level debugging, 0 messages logged, xml disabled,
                     filtering disabled
    Buffer logging: level debugging, 88 messages logged, xml disabled,
                    filtering disabled
    Logging Exception size (8192 bytes)
    Count and timestamp logging messages: disabled
No active filter modules.
    Trap logging: level informational, 94 message lines logged
Log Buffer (8192 bytes):
Oct 28 20:41:05.025 UTC: %SYS-5-CONFIG_I: Configured from console by console
Oct 28 20:41:05.029 UTC: %HA_EM-5-LOG: eemCfgEvLog: Configuration event detected
Oct 28 21:09:29.120 UTC: %SYS-5-CONFIG_I: Configured from console by console
Oct 28 21:09:29.120 UTC: %HA_EM-5-LOG: eemCfgEvLog: Configuration event detected
router1#
```

The custom Syslog message is now issued whenever a Cisco IOS Software Syslog message matching the string "%SYS-5-CONFIG_I: Configured" is issued. While this example may not be very useful, it does illustrate the procedure for configuring an EEM applet action. More useful adaptations might utilize the optional parameters associated with the **`event syslog`** command. For instance, the optional **[occurs num-occurrences]**, **[period period-value]**, and **[priority priority-level]** parameters can be used to issue a higher priority message after several Syslog messages have been issued in quick succession.

Environment variables can be used as output in Syslog messages to further distinguish the trigger source details. Table 2 lists the available environment variables.

Table 2. Environment Variables Available for Use

Environment Variables Available for All Events	
$_event_type	The event type that triggered the event.
$_event_pub_time	The time at which the event type was published.
Environmental Variables Available for SNMP Events	
$_snmp_oid	The Simple Network Management Protocol (SNMP) object ID that caused the event to be published.
$_snmp_oid_val	The SNMP object ID value when the event was published.
Environment Variables Available for Syslog Events	
$_syslog_msg	The syslog message that caused the event to be published.

These variables can be used in the '**msg'** text and will be replaced with the relevant text.

CREATING POLICIES USING TCL

More extensive policies can be created using the script policy engine. Cisco IOS Software EEMv2 incorporates a Tcl-based policy engine compatible with Tcl 8.3.4. Tcl-based policies can be developed that interact with Cisco IOS Software via CLI commands and a set of environment variables.

A Tcl policy, when registered, becomes an event subscriber. Once a registered event is detected, the EEM server will trigger any and all corresponding event subscribers interested in this particular event.

A Tcl-based policy is then invoked and run to completion. The Tcl policy executes within a framework established by Cisco IOS Software. A user-written Tcl-based policy will not disrupt other Cisco IOS Software. Even if a programming error in the policy resulted in an infinite loop, the Cisco IOS Software execution framework would prevent critical Cisco IOS Software functions from harm or degradation. However, the policy can and will impact Cisco IOS Software operation and packet forwarding as dictated by the programming of the policy. In other words, be aware that these policies can cause unintended results if there are errors in the action logic of the policy.

Tcl-based policies are created following the same basic steps as described previously for EEM Version 1.0. The difference being that policies are registered and event triggers are defined using Tcl keywords. The steps to implement a Tcl-based policy are:

Step 1. Determine the purpose of the policy and program the policy

It is important to determine exactly what needs to be accomplished with the policy before implementation. Make sure it is known exactly what needs to be achieved and the effect policy will have on the network. Use standard programming techniques such as flowcharting, code walk-throughs, and design reviews for advanced policies.

Step 2. Establish event trigger

The start of every script-based policy must register as a subscriber for an event trigger using the **event_register** keyword. This defines the event that triggers the policy and schedules execution of the policy. The following event register keywords are available:

EEM event registration keywords

```
event_register_appl
event_register_process
event_register_timer_subscriber
event_register_timer
event_register_counter
event_register_interface
event_register_ioswdsysmon
event_register_rf
event_register_snmp
event_register_syslog
event_register_wdsysmon
```

Step 3. Get event details

Use the **event_reqinfo** keyword, if desired, to obtain details about the event that has triggered the policy.

Step 4. Define action

Define the action to take upon detection of the event as established in Step 2. This is accomplished using one of the action keywords such as:

```
action_process
action_syslog
action_reload
action_script
action_program
action_snmp_trap
```

There are many other EEM policy keywords and extensions to the Tcl language that facilitate the definition of practically to any policy action imaginable. For a complete list of the EEM policy keywords and programming techniques, refer to the document *Writing Embedded Event Manager Policies* in the reference section of this document.

Getting Started

To get started writing EEM Tcl policies, first define the policy directory.

```
router2#mkdir ABCCoTclPol
Create directory filename [ABCCoTclPol]?
Created dir disk0:ABCCoTclPol
router2#dir
Directory of disk0:/
    1  drw-           1  Oct 26 2003 13:37:42 +00:00  sys
    6  drw-           1  Oct 30 2003 12:56:04 +00:00  ABCCoTclPol
47843328 bytes total (29356032 bytes free)
router2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
```

```
router2(config)#event manager directory user policy disk0:/ABCCoTclPol
router2(config)#^Z
router2#router2#sh event man dir user policy
disk0:/ABCCoTclPol
router2#
```

Then, copy the Tcl policy scripts to this directory.

```
router2#copy tftp disk0:
Address or name of remote host []? 10.1.88.9
Source filename []? sl_cfgSaveRemT.tcl
Destination filename [sl_cfgSaveRemT.tcl]? ABCCoTclPol/sl_cfgSaveRemT.tcl
Accessing tftp://10.1.88.9/sl_cfgSaveRemT.tcl...!
1232 bytes copied in 0.620 secs (1987 bytes/sec)
router2#dir
Directory of disk0:/
    1  drw-           1  Oct 26 2003 13:37:42 +00:00  sys
    6  drw-           1  Oct 30 2003 12:56:04 +00:00  ABCCoTclPol
47843328 bytes total (29351936 bytes free)
router2#cd ABCCoTclPol
router2#dir
Directory of disk0:/ABCCoTclPol/
    8  -rw-        1232  Oct 30 2003 14:14:58 +00:00  sl_cfgSaveRemT.tcl
47843328 bytes total (29351936 bytes free)
router2#
```

Once the policy has been saved, register it using the configuration command shown below:

```
router2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
router2(config)#event manager policy sl_cfgSaveRemT.tcl type user
router2(config)#
```

Verify the policy is registered using the command:

```
router2#sh event manager policy registered
No.  Type    Event Type          Trap  Time Registered           Name
1    user    syslog              Off   Thu Oct30  14:54:17 2003  sl_cfgSaveRem.tcl
 occurs 1 pattern {%SYS-5-CONFIG_I: Configured}
 nice 0 priority normal maxrun 90.000
router2#
```

Note: If replacing a policy with a new version, re-register the policy to pick up the new version. Do this by using the ‘no’ form of the event manager policy command. For instance, **no event manager policy sl_cfgSaveRemT.tcl type user** followed by **event manager policy sl_cfgSaveRemT.tcl type user**.

Example 1 illustrates a simple Tcl-based EEM policy. This sample policy is for explanatory purposes. Its function may not be useful in a real network but it does illustrate a few points. For more details about writing Tcl-based EEM policies, refer to the document, Writing Embedded Event Manager Policies in the reference section of this document.

Example 1: Tcl-based EEM Policy

```
::cisco::eem::event_register_syslog occurs 1 pattern “\%SYS-5-CONFIG_I: Configured” maxrun_sec 90
#
# namespace imports
#
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
#
# Body of policy
#
# check for env variable u_cfgSave_on—if it exists, use the value
# else set it to 0 = off
#
if {![info exists u_cfgSave_on]} {
     #u_cfgSave_on is an option, must set to 0 if not set.
     set u_cfgSave_on 0
}
#
if $u_cfgSave_on==1 {
    action_syslog msg “Config save mode set, saving configuration to nvram”
    if [catch {cli_open} result] {
        error $result $errorInfo
    } else {
        array set cli $result
    }
    if [catch {cli_exec $cli(fd) “enable”} result] {
        error $result $errorInfo
    }
    if [catch {cli_write $cli(fd) “copy run start”} result] {
        error $result $errorInfo
    }
    if [catch {cli_read_pattern $cli(fd) “Destination filename”} result] {
        error $result $errorInfo
    }
    if [catch {cli_write $cli(fd) “\n”} result] {
        error $result $errorInfo
    }
    if [catch {cli_read_drain $cli(fd)} result] {
        error $result $errorInfo
    } else {
        set cmd_output $result
```

```
  41            action_syslog msg “copy command reponse—$cmd_output”
  42        }
  43        if [catch {cli_close $cli(fd) $cli(tty_id)} result] {
  44            error $result $errorInfo
  45        }
  46    } else {
  47        action_syslog msg “Config save mode not set, remember to save your configuration
to nvram”
  48    }
  49    action_syslog msg “done”
  50    exit 0
```

Referring to Example 1, line 1 shows how the policy is registered. This policy is triggered by a Syslog message matching the regular expression pattern “\%SYS-5-CONFIG_I: Configured”.

Lines 5 and 6 define the namespaces that include the Cisco IOS Software Tcl extensions and library commands.

Line 13 checks that the required environment variable exists or, if not, sets it to 0. For the purposes of this EEM policy, the environment variable “u_cfgSave_on” is used to enable and disable this policy via a configuration command.

Setting “u_cfgSave_on” by:

```
router2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
router2(config)#event manager environment u_cfgSave_on 1
router2(config)#
```

This enables the policy, since this global variable is checked on line 18.

Line 19 illustrates the command used to write out a Syslog message from a policy. This line results in the following Syslog message:

```
Feb 17 17:24:56 UTC: %HA_EM-6-LOG: sl_cfgSaveRem.tcl: Config save mode set, saving configuration
to nvram
```

Lines 20 through 45 show how the CLI is controlled from an EEM policy. On line 20, the CLI is “opened”. Thereafter, the cli file descriptor that was returned is used as a reference for input/output (I/O) to the CLI. An internal user id and password are used, so there is no need to supply one for policy interaction.

On line 25 the command is issued to enter privileged exec mode.

On line 28, the copy running-config startup-config command is issued to save the configuration to nvram.

When a copy command is entered via the CLI, interaction such as the following occurs:

```
router2#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
router2#
```

Lines 31-40 deal with that interaction. Line 31 waits for the string “destination filename” and line 34 issues a newline in response.

Lines 40 and 41 capture the output and create a Syslog message to confirm the result.

Again, this Tcl policy is merely presented to illustrate some of the capabilities. Cisco IOS Software capabilities for configuration archiving and rollback offer a better alternative than creating a policy as a reminder to save the configuration. However, our purpose in analyzing a simple policy is served.

USING ENVIRONMENT VARIABLES

The Cisco IOS Software EEM environment variables are global variables that are defined external to the policy before the policy is run. Environment variables offer an effective way to set parameters used within a policy. The policy can check the environment variables during initialization and use the settings to determine specific actions to take or to customize an otherwise more generic policy. Environment variables may be set using the configuration command:

```
event manager environment varname varvalue
```

All Cisco IOS Software EEM environment variables begin with "_" by convention. In order to avoid future conflict, customers are urged not to define new variables that start with "_".

The Cisco IOS Software EEM environment variables set can be displayed on the system with the privileged EXEC mode command:

```
show event manager environment [all|varname]
```

RELATED FEATURES AND FURTHER DEVELOPMENT

Cisco IOS Software EEM opens the door to a distributed, cooperative model for network management where the devices being managed take an active role in the process. The ability to program local actions, adapt immediately to changing conditions and events, and employ the network intelligence inherent in Cisco IOS Software will beget more resilient network services.

Cisco IOS Software EEM will be increasing its event coverage with subsequent releases: new event detectors will be added over time, as the scope and breadth of the events that can be handled increases. Integration with other subsystems, particularly in the area of diagnostics is essential to detect errors, faults, or failures and prevent them from resulting in network service disruption.

Recently Added Event Detectors

Cisco is developing a number of new event detectors. One important addition planned is the integration with the Generic Online Diagnostic (GOLD) subsystem. Proactive online diagnostic and health monitoring is an essential internal service and can serve as triggers for High Availability switchover. Without proactive fault detection, the ability to switch traffic to an alternate path may be delayed until the failure has already disrupted network service. Enhancing internal diagnostic systems with a programmable interface provides an invaluable tool for support staff and Cisco Customer Advocacy and Technical Assistance Center (TAC) engineers. The GOLD subsystem is being integrated with EEM. This hardware-independent, diagnostic subsystem will be available across various Cisco products offering consistent function and CLI. The diagnostic subsystems provide a framework for:

- Boot up diagnostics—during boot up and card Online Insertion or Removal (OIR)
- Health monitoring diagnostics including execution while system is in operation
- On-demand diagnostics
- Scheduled diagnostics
- Call-home capability integrated with Cisco Network Services Event Bus.
- Integration to Cisco IOS Software EEM

Another event detector under consideration is an interface to external devices and sensors. Investigations are underway to determine the applicability for integration with Uninterruptible Power Supplies (UPS), external environmental monitors, security systems, Global Positioning Systems (GPS), etc.

Still another event detector which was added in Release 12.4(2)T is an object tracking event detector. Cisco IOS Software includes another subsystem derived from the first hop routing protocol tracking capability that began with Hot Standby Routing Protocol (HSRP). Subsequent to the interface tracking feature debut within the HSRP subsystem, an independent subsystem and feature known as Enhanced Object Tracking (EoT) has been delivered. EoT extends the interface tracking capability of HSRP by allowing additional objects to be tracked. HSRP, Gateway Load Balancing Protocol (GLBP), and Virtual Router Redundancy Protocol (VRRP) subsystems become ‘clients’ of EoT and benefit from the new capabilities. EoT has been integrated with the Cisco IOS Service Assurance Agent (SAA), increasing the span of objects that may be tracked. Any Cisco IOS SAA operation is now a candidate for tracking. Cisco IOS Software EEM and EoT are synergistic, they both offer tracking of events or ‘objects’. EEM and EoT will be integrated to allow EoT to be notified of events seen by EEM event detectors. Likewise, an event detector for EoT tracked objects is also being considered.

Watch for details on these and other event detectors in future Cisco IOS Software releases.

REFERENCES

- Cisco IOS Software Documentation
  - Embedded Event Manager 2.1: http://www.cisco.com/en/US/products/ps6350/products_configuration_guide_chapter09186a008045578a.html
  - Embedded Event Manager 2.2: http://www.cisco.com/en/US/products/ps6441/products_feature_guide09186a00804aae8c.html
  - Writing Embedded Event Manager Policies: http://www.cisco.com/en/US/products/sw/iosswrel/ps5207/products_feature_guide09186a008041231a.html
- Tcl Documentation: http://tcl.activestate.com/doc/, http://tcl.activestate.com/man/tcl8.3/

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

European Headquarters
Cisco Systems International BV
Haarlerbergpark
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands
www-europe.cisco.com
Tel: 31 0 20 357 1000
Fax: 31 0 20 357 1100

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
www.cisco.com
Tel: 408 526-7660
Fax: 408 527-0883

Asia Pacific Headquarters
Cisco Systems, Inc.
168 Robinson Road
#28-01 Capital Tower
Singapore 068912
www.cisco.com
Tel: +65 6317 7777
Fax: +65 6317 7799

Cisco Systems has more than 200 offices in the following countries and regions. Addresses, phone numbers, and fax numbers are listed on **the Cisco Website at www.cisco.com/go/offices.**

Argentina • Australia • Austria • Belgium • Brazil • Bulgaria • Canada • Chile • China PRC • Colombia • Costa Rica • Croatia • Cyprus Czech Republic • Denmark • Dubai, UAE • Finland • France • Germany • Greece • Hong Kong SAR • Hungary • India • Indonesia • Ireland • Israel Italy • Japan • Korea • Luxembourg • Malaysia • Mexico • The Netherlands • New Zealand • Norway • Peru • Philippines • Poland • Portugal Puerto Rico • Romania • Russia • Saudi Arabia • Scotland • Singapore • Slovakia • Slovenia • South Africa • Spain • Sweden • Switzerland • Taiwan Thailand • Turkey • Ukraine • United Kingdom • United States • Venezuela • Vietnam • Zimbabwe

205393.BA_ETMG_PI_12.05

Printed in the USA