White Paper

Embedded Event Manager (EEM) on the Cisco Catalyst 6500 Series

1. INTRODUCTION

With Cisco® IOS® Software Release 12.2(18)SXF4, Cisco IOS Software with Software Modularity became a reality. While the Software Modularity aspect is the most obvious enhancement of this release, there is also another key feature in this software version, which could enhance and revolutionize the scope of operational management in customer's networks for the Cisco Catalyst® 6500 Series.

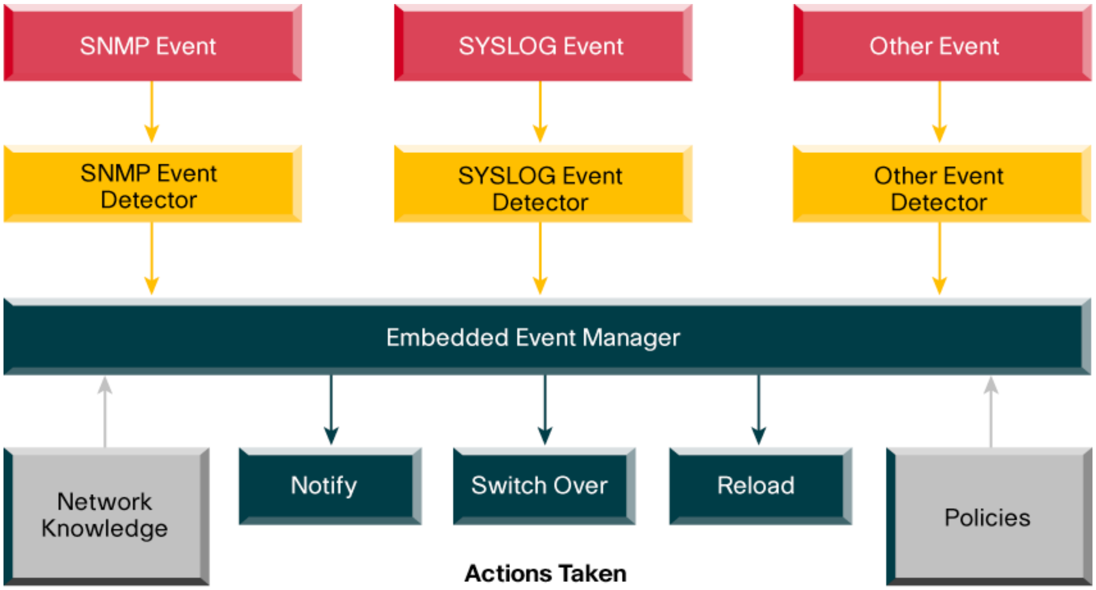
Embedded Event Manager (EEM) is a policy-based framework that provides a way to monitor key system events and then act on those events through a set policy. The policy is, quite simply, a preprogrammed script loaded by the administrator, which defines actions that the switch should invoke based on set events occurring. The script can generate actions, including, but not limited to, generating custom SYSLOG or SNMP traps, invoking CLI commands, forcing a failover, and much more.

This paper will provide an insight into EEM as it is architected on the Catalyst 6500.

2. WHAT IS EEM, AND HOW CAN IT BE USED?

EEM is a flexible programmable policy based framework that allows an administrator to customize a script to invoke an action based on a given set of events occurring. The basic makeup of the EEM facility on the Catalyst 6500 is shown in Figure 1.

Figure 1. Basic EEM Architecture



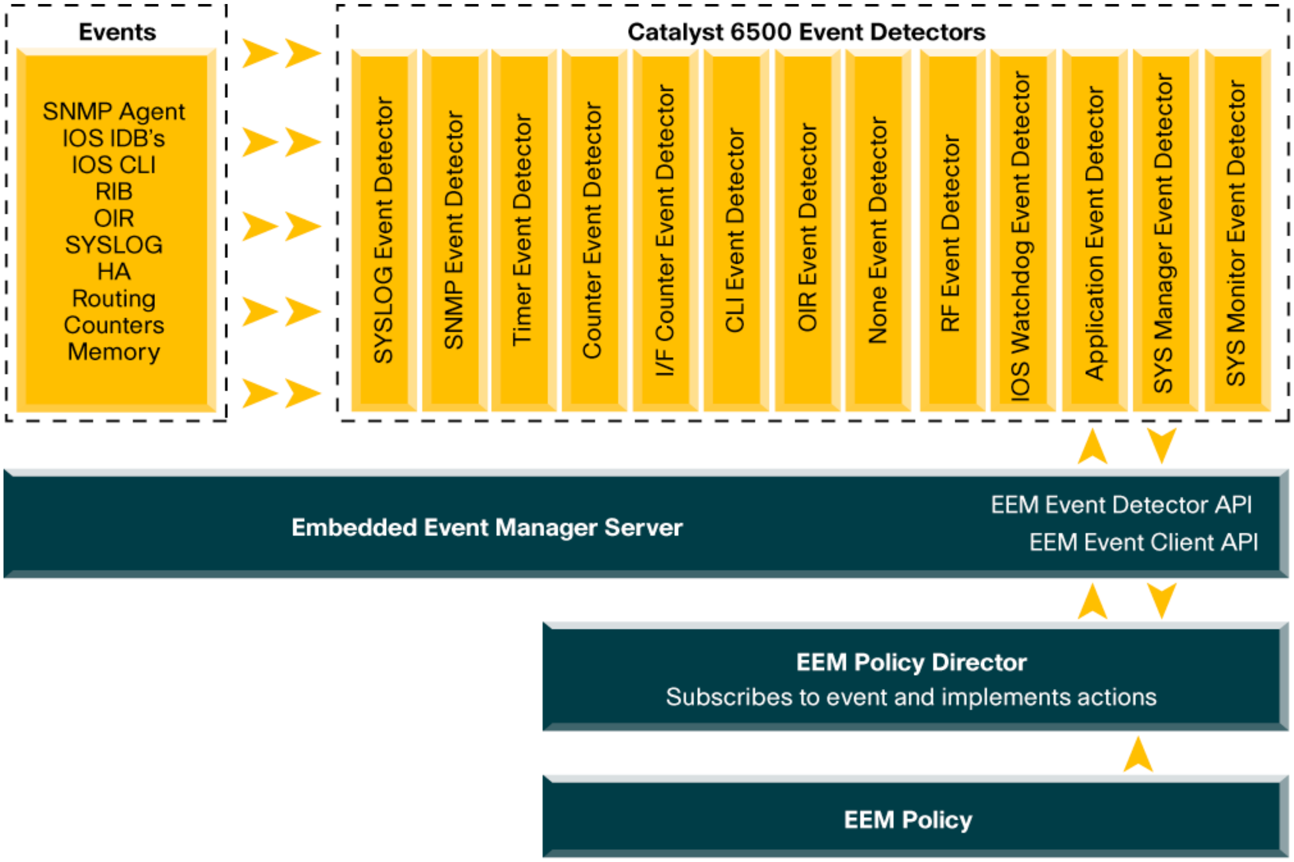
The essence of how EEM operates is summarized as follows. A system event occurs that is picked up by an event detector. What is an event? It could be when a specific SYSLOG message is generated, or when a certain command is executed on the CLI, or if a given counter exceeds a set threshold, when a line card is inserted, or when an SSO failover is initiated. In fact, these are just a few examples of the many events that can be generated. EEM incorporates a number of event detectors, which are subsystem processes designed to monitor the system for those key events. The given event detector alerts the EEM subsystem and passes relevant information to it regarding the event. A predefined script that is created by the administrator and registered with EEM is started, which will use the event information to invoke a given action on the switch.

EEM provides for two types of scripts that can be used as the criteria for invoking actions based on given events occurring. A script that is entered via the CLI is one form of script supported by EEM. This form of script is known as an applet. The second form of script supported is a TCL script. Support for TCL in the Catalyst 6500 EEM subsystem is based on TCL v8.3.4. This is the same version of TCL used for TCL Shell in Cisco Router IOS Software. While applet-based scripts provide an easy option from which to load a script onto the switch, it is with TCL where the more flexible (and powerful) scripts can be developed. Both script options are discussed in more detail later in this paper.

3. EEM ARCHITECTURE ON THE CATALYST 6500

The version of EEM used on the Catalyst 6500 Cisco IOS Software Release 12.2(18)SXF4 is based on EEM V2.1.5. This version combines a number of elements that make up the entire EEM feature. Each of the EEM elements found on the Catalyst 6500 is detailed in Figure 2.

Figure 2. EEM Detailed Architecture View



This version of EEM initially provides support for 13 event detectors, although more event detectors will become available with future releases of the Catalyst 6500 Cisco IOS Software. Each event detector is a separate subsystem in itself and is responsible for interfacing between the publisher of an event and the Event Manager Server. Each of the supported event detectors in this initial Cisco IOS Software release is detailed in the following list.

- **Application Event Detector:** Administrator configured policies registered to the EEM subsystem can publish their own events using this event detector; this gives a policy the ability to trigger another policy to execute.
- **CLI Event Detector:** When a CLI command is entered from the console that matches a pre defined CLI command defined by the administrator, then this event detector can generate an event. This event typically uses a pattern match to look for the specific command in order to trigger an event
- **Counter Event Detector:** Should the value of a designated counter identified within a policy change, then this event detector can generate an event. An example is policy “A” increments a counter, and when that counter exceeds a threshold then policy “B” is invoked.
- **Interface Counter Event Detector:** When a threshold (absolute or incremental) for a specific port counter is crossed, then this event detector can generate an event. This provides an easier way to track interface statistics. The interface counters that are supported include:
    - Input Errors
    - Input Errors CRC
    - Input Errors Frame
    - Input Errors Overrun
    - Input Packets Dropped
    - Interface Resets
    - Output Buffer Failures
    - Output Buffer Swapped Out
    - Output Errors Underrun
    - Output Errors
    - Output Packets Dropped
    - Receive Broadcasts
    - Receive Giants
    - Receive Rate PPS
    - Receive Rate BPS
    - Receive Runts
    - Receive Throttle
    - Reliability
    - RX Load
    - TX Load
    - Transmit Rate PPS
    - Transmit Rate BPS

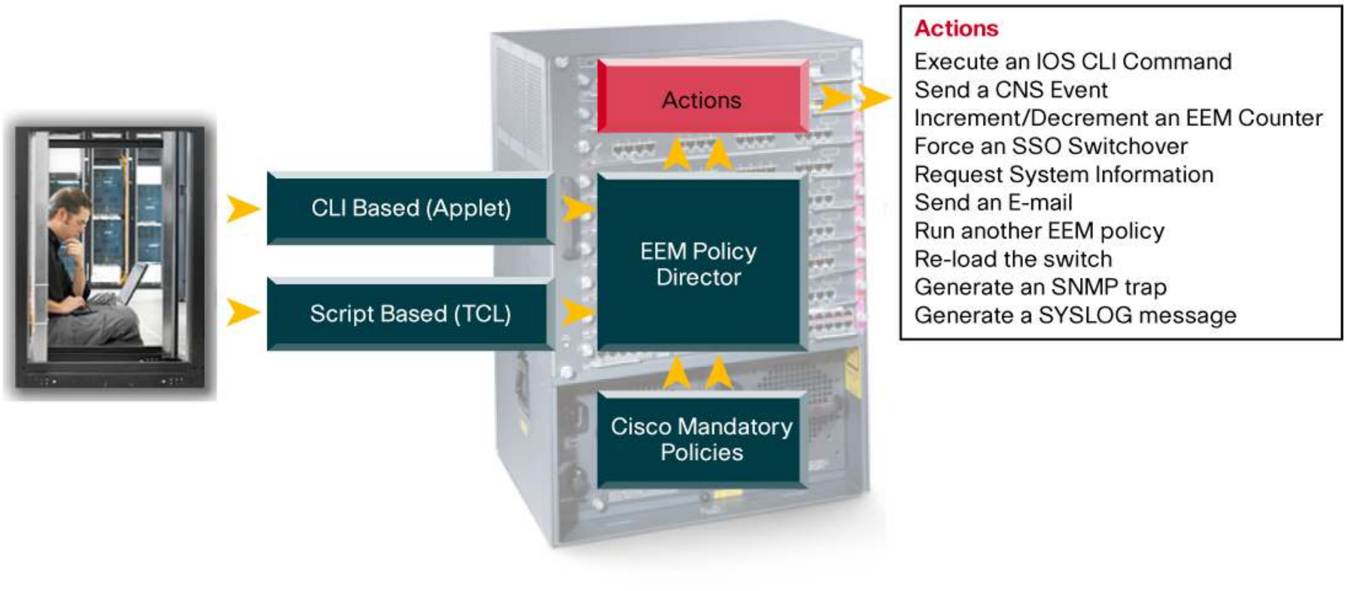
- **Cisco IOS Software Watchdog Event Detector:** This event detector publishes an event when one of the following occurs:
    - CPU Utilization for an Cisco IOS Software process crosses a threshold
    - Memory use for an Cisco IOS Software process crosses a threshold
    - Total available system memory crosses a threshold
    - Total used system memory crosses a threshold
    - Total system CPU crosses a threshold
- **System Monitor Event Detector:** Should an Cisco IOS Software memory leak occur or a deadlock or loop occur in an Cisco IOS Software task (that is, an Cisco IOS Software modular process), this detector will generate an event.
- **None Event Detector:** This event detector is used as a placeholder for policies that are manually triggered through the "event manager run" command on the switch CLI.
- **OIR Event Detector:** This event detector will monitor the system for hardware (such as line cards and so on) that are inserted or removed and, should this occur, generate an event.
- **Redundancy Framework Event Detector:** Hardware or software high availability events related to an SSO failover, or any redundancy framework state transition will cause this event detector to generate an event.
- **SNMP Event Detector:** Allows an SNMP object to be polled at a regular interval, and when the value of the object matches a specified value, an event is generated.
- **System Manager Event Detector:** When a Cisco IOS Software modularity process is stopped (normally or abnormally) or is started, then this event detector will generate an event.
- **SYSLOG Event Detector:** This event detector will generate an event when a set SYSLOG message is generated. Regular Expressions can be used to match on part of a SYSLOG message to generate the event. This detector also allows a match on a number of patterns matching before generating an event (for example, if SYSLOG message x occurs within 5 minutes, then generate an event).
- **Timer Event Detector:** Used to generate an event based on one of the following four timer events:
    - An absolute time of day timer
    - A countdown timer that publishes an event when the value hits 0
    - A watchdog timer that publishes an event when the timer counts to 0, upon which it resets itself and begins the cycle again
    - A CRON timer that uses a UNIX-based CRON specification to indicate when an event should be published

The EEM Server performs the role of the central information repository for EEM managing event registration and policy execution. It provides a means to handle event requests such as the creation, registration, and removal of events, as well as an ability to store persistent data that can be referenced by an active and running policy. It also offers a way for multiple policies to interface with one another, allowing them exchange data.

As a server, it incorporates two APIs (application programmable interfaces). One of these APIs ties into the event detector processes, allowing it to communicate with each of the detectors when an event is registered on the system. The second API is used to communicate with the EEM policy director, which is the repository for registered policies.

The EEM policy director is a process that is used to accept and register user built scripts to the EEM subsystem and is summarized in Figure 3.

Figure 3. EEM Policy Director



Finally, there are the EEM policies themselves. We have already identified two forms of polices, those being the applets and TCL scripts. There are two types of TCL scripts, namely user written scripts and Cisco Systems® written TCL scripts (otherwise known as a mandatory policy) which are embedded within the Cisco IOS Software image. Mandatory policies are automatically enabled on system startup; however, a specific mandatory policy can be individually disabled form the switch CLI. Both the TCL scripts and the applets have a set number of actions that they can invoke. These actions include the following.

- Execute an Cisco IOS Software CLI command and receive the result
- Send a CNS event
- Log a message to SYSLOG
- Send an e-mail or system page
- Increment or decrement an EEM counter
- Force a failover to the SSO (Stateful Switchover) standby supervisor
- Request system information
- Invoke another EEM policy to be started
- Reload the switch
- Send an SNMP trap with custom data
- Publish an application specific EEM event
- Run a TCL script

The following sections will provide more detail on adding applets and TCL scripts to the switch.

4. EEM APPLETS

While the applets functionality does not match what a TCL script can do, it does provide a simple way for a policy to be created from the switch CLI and registered with the switch.

The applet is initially created using the following command:

```
[no] event manager applet <applet-name>
```

For an applet (or TCL script) to function, it must be registered with the EEM policy director. The notion of creating an applet from the CLI using the above command also inherently registers the applet with EEM at the same time. Once this command is entered, the system moves the user into applet configuration mode, where additional applet commands can be entered. There are essentially three configuration commands that can then be entered. These include the following.

```
[no] event <event-type>
[no] action <label> <action-type>
[no] set <label> <var-name> <value>
```

For any configured applet only one “event” command can be entered. The event command identifies the event detector that this applet is working with. If there is no “event” command configured, a warning message is posted when exiting the event configuration mode, and the applet is not registered. The action statement indicates the action that should be invoked should there be a match to the configured event. The command can be used to set environment variables that might be referenced in this applet. Environment variables are discussed later in this paper.

Let us take a look at how an applet can be configured. The first task is to define an applet in configuration mode as follows:

```
C6500(config)#event manager applet my_applet
C6500(config-applet)#
```

This command creates an applet called “my_applet” and registers it with the EEM policy director. It also places the user into applet configuration mode where subsequent configuration commands can be entered. The next step of this applet configuration requires an event command to be configured. The event command options available are shown in the following command output.

```
C6500(config-applet)#event ?
  Application  Application specific event
  Cli          CLI event
  Counter      Counter event
  Gold         GOLD event
  Interface    Interface event
  Ioswdsysmon  IOS WDSysMon event
  None         Manually run policy event
  Oir          OIR event
  Process      System Manager event
  Rf           Redundancy Facility event
  Snmp         SNMP event
  Syslog       Syslog event
  Timer        Timer event
  Wdsysmon     WDSysMon event
```

For this example, we will use the CLI event detector to look for a pattern match on a CLI command as shown in the following example.

```
C6500(config-applet)#event cli pattern "conf t" ?
  occurs  The number of occurrences before raising the event
  period  Number of occurrences must occur within this time period
  skip    Whether to skip CLI command execution
  sync    CLI and EEM policy execution sync or async
C6500(config-applet)#event cli pattern "conf t" sync ?
  No      Policy and CLI will run asynchronously
  Yes     Run policy and the result determines whether to run CLI
C6500(config-applet)#event cli pattern "conf t" sync no ?
  Occurs  The number of occurrences before raising the event
  Period  Number of occurrences must occur within this time period
  Skip    Whether to skip CLI command execution
C6500(config-applet)#event cli pattern "conf t" sync no skip ?
  No      CLI command should be executed
  Yes     CLI command should not be executed
C6500(config-applet)#event cli pattern “conf t” sync no skip no
```

This command calls the CLI event detector to monitor the CLI and look for a command that matches the pattern “conf t.” This command is used when an administrator enters configuration mode.

Once an event command is configured, a corresponding action command is required to be configured to determine what happens when this applet detects the string “conf t” on the CLI. The available actions are shown in the following CLI output.

```
C6500(config-applet)#action 1.0 ?
  Cli                Execute a CLI command
  cns-event          Send a CNS event
  counter            Modify a counter value
  force-switchover   Force a software switchover
  info               Obtain system specific information
  mail               Send an e-mail
  policy             Run a pre-registered policy
  publish-event      Publish an application specific event
  reload             Reload system
  snmp-trap          Send an SNMP trap
  syslog             Log a syslog message
```

In our example we are simply going to put a message on the console using SYSLOG as follows:

```
C6500(config-applet)#action 1.0 syslog msg "Configuration by Authorized personnel ONLY"
```

Should the word “conf t” be detected on the CLI, a SYSLOG message will be generated alerting the user that only authorized personnel are allowed to enter configuration mode. It is worth noting that the “1.0” is a label used to uniquely identify this action command. If there were a requirement to configure multiple actions for this event, then the subsequent action commands would need a unique label as in the following example.

```
C6500(config-applet)#event cli pattern “conf t” sync no skip no
C6500(config-applet)#action 1.0 syslog msg etc etc
C6500(config-applet)#action 2.0 etc etc
C6500(config-applet)#action 3.0 etc etc
```

The administrator would then exit out of applet configuration mode. The applet is added to the current switch configuration and can be viewed as follows:

```
C6500#show run | begin event
event manager applet my_applet
 event cli pattern "conf t" sync no skip no
 action 1.0 syslog msg "Configuration by Authorized Personnel ONLY"
!
end
```

EEM also provides a way to confirm that the entered applet has been registered and correctly entered. This is shown in the following show command set.

```
C6500#show event manager policy registered
No.  Class   Type    Event Type Trap  Time Registered          Name
1    applet  system  cli        Off   Thu Jan 12 02:43:57 2006  my_applet
 pattern {conf t} sync no skip no
 action 1.0 syslog msg "Configuration by Authorized Personnel ONLY"
```

Finally, the applet can be seen in action when an administrator enters configuration mode using the “conf t” command set.
This is shown below.

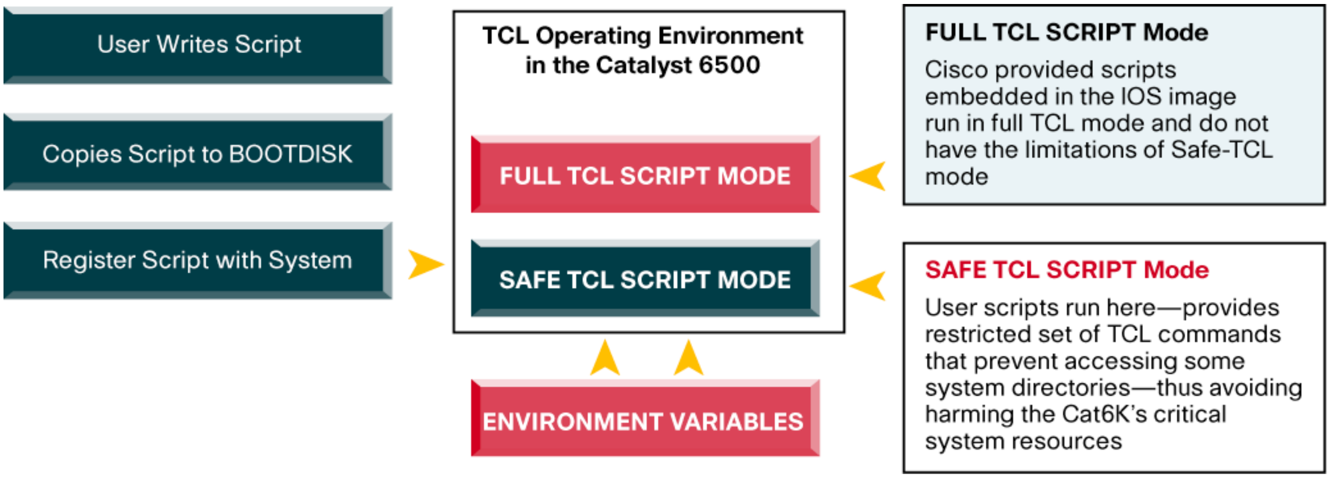
```
C6500#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C6500(config)#
3w2d: %HA_EM-6-LOG: my_applet: Configuration by Authorized Personnel ONLY
```

5. EEM TCL SCRIPTS

While applets provide for a simple and effective method for adding basic scripts to the system, it is with TCL scripts where the true power and flexibility of EEM become evident. TCL (or Tool Control Language) is a string-based command language that is interpreted at runtime (in much the same way as the “BASIC” programming language), rather than being compiled in a traditional programming sense. The EEM subsystem support for TCL is based on TCL v8.3.4 and contains the full complement of commands available with that release along with a number of TCL command extensions designed specifically for the Catalyst 6500 system.

The EEM architecture incorporates two operational levels within with a TCL script can run. These levels, in many respects, provide a mechanism to protect the switch from user-based scripts inadvertently accessing system resources that could override the integrity of a running system. Cisco mandatory scripts run in what is referred to as full TCL mode. This mode provides full access to all of the switch’s resources. User-built scripts, however, run in what is referred to as safe TCL mode. (See Figure 4.)

Figure 4. TCL Execution Modes on the Catalyst 6500


User Writes Script
Copies Script to BOOTDISK
Register Script with System
TCL Operating Environment in the Catalyst 6500
FULL TCL SCRIPT MODE
SAFE TCL SCRIPT MODE
ENVIRONMENT VARIABLES
FULL TCL SCRIPT Mode
Cisco provided scripts embedded in the IOS image run in full TCL mode and do not have the limitations of Safe-TCL mode
SAFE TCL SCRIPT Mode
User scripts run here—provides restricted set of TCL commands that prevent accessing some system directories—thus avoiding harming the Cat6K’s critical system resources

The “safe TCL script” mode of operation runs the script inside a “safe interpreter,” isolating it from other applications. The execution of scripts in safe TCL mode is under the control of a master interpreter allowing it to control the service requests made by the running script. The safe TCL mode allows Cisco to disable or customize individual TCL commands, thus providing a means to protect the system from a runaway script. This mode also allows the administrator to totally disable user-based TCL scripts from the CLI using a single command. It is possible to modify a Cisco mandatory policy, but doing so requires the user to move the modified policy to the user directory and run it in safe TCL mode.

Furthermore, an additional level of security is implemented for those scripts that invoke a CLI command. EEM provides a command that allows the specification of the Cisco IOS Software user ID, allowing a TACACS+ command authorization service to be used.

Environment variables are commonly used by this TCL scripting feature. An environment variable is a global variable that is set outside of the TCL script, but one that can be referenced from within the script. These variables provide a useful vehicle for the script to learn more about the system within which it is operating and to learn more about the environment that triggered an event. There are a couple of types of environment variables that exist and these include:

- User-defined environment variables that are defined by a user
- Cisco defined environment variables that are either defined by Cisco or created for a specific policy

The **event manager environment** command is used to set environment variables on the switch. All Cisco defined environment variables start with an underscore (“_”). This is a reserved character and cannot be used by users when defining their own variables. There are a number of Cisco defined environment variables available, all of which are documented in the Cisco IOS Software manuals on Cisco.com, but a few of these are listed in Table 1 for the purposes of giving examples of what information they can provide. This is not a complete list of variables available. Please refer to the EEM documentation on Cisco.com for a more exhaustive list.

Table 1. Example Environment Variables

Script Action	Environment Variable	Variable Purpose
Want to send an e-mail from within a script	***_email_server***	Used to identify the IP address of the SMNP server used when e-mails are sent from within a script
	_email_to	Used to identify the recipient of the sent e-mail
	_email_from	Used to identify the originator of the sent e-mail
Inspect the counter event detector	***_counter_name***	Inspect the name of the counter
	_counter_value	Check the reference point value of the counter
Check an SNMP MIB object	***_snmp_oid***	Identify the MIB object being interrogated
	_snmp_oid_value	Contains the value of the SNMP MIB object

Environment variables that have been set on the switch can be interrogated using the following command.

```
C6500#show event manager environment all
No.  Name                           Value
1    _crash_reporter_url            http://165.22.30.31/cr/interf
2    _email_server                  10.46.11.23
3    _email_from                    eem@cisco.com
4    _email_to                      techo@cisco.com
5    pingcheck_freq                 30
6    pingcheck_addr                 10.1.1.1
<..>
```

When building a TCL script, it is important to note that while any of the TCL script commands can be used within the script, the script itself *must* start off with an **event_register** command. This command is used to tell the EEM server what event is used to trigger the policy. The available commands for this purpose are listed below.

- event_register_apl
- event_register_cli
- event_register_counter
- event_register_interface
- event_register_ioswdsysmon
- event_register_none
- event_register_oir
- event_register_process
- event_register_rf
- event_register_snmp
- event_register_syslog
- event_register_timer
- event_register_wdsysmon

An example of using this within a script is shown below.

```
::cisco::eem::event_register_cli occurs 1 pattern “conf t"
```

This example calls on the CLI event detector to look for a single instance (identified by the “occurs 1” string) of the “conf t” command. Following this command, standard TCL commands apply and can be written to invoke an action for that given event.

Once the TCL script has been written, it must be loaded on the switch and registered with the EEM policy director. Prior to loading it on the switch, a directory must be created on the switch file system, and this directory must also be registered with the EEM policy director. An example of this is shown with the following command example.

```
C6500#mkdir USER_TCL
Create directory filename [USER_TCL]?
Created dir disk0:USER_TCL

C6500#dir disk0:
Directory of disk0:/
    1  drw-           1  Oct 18 2005 01:14:07 +00:00  USER_TCL
47843328 bytes total (47843327 bytes free)
```

Now that the directory has been created, it must be registered with the EEM policy director. This is achieved using the **event manager directory** command as follows.

```
C6500#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C6500(config)#event manager directory user policy disk0:/USER_TCL
C6500(config)#^Z
C6500#show event man dir user policy
disk0:/USER_TCL
```

The directory is registered with this command and is now ready for TCL scripts to be loaded into it. The process of loading, registering and executing the script is shown in the following sequence of commands. First the TCL script is copied onto the switch as follows.

```
C6500#copy tftp disk0:
Address or name of remote host []? 10.66.240.46
Source filename []? sample.tcl
Destination filename [cfgSave.tcl]? USER_TCL/sample.tcl
Accessing tftp://10.66.240.46/sample.tcl...!
1232 bytes copied in 0.620 secs (1987 bytes/sec)
C6500#
```

After copying the TCL script to the switch, it must be registered to the EEM policy director with the **event manager policy** command as follows.

```
C6500#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
C6500(config)#event manager policy sample.tcl type user
```

Confirmation that the TCL script has been registered can be achieved using the following show command.

```
C6500 #show event manager policy registered
No.  Type    Event Type          Trap  Time Registered           Name
1    user    cli                 Off   Thu Jan12  15:18:16 2006  sample
 occurs 1 pattern
 nice 0 priority normal maxrun 90.000
```

The script is now loaded, registered, and primed for execution when the respective event detector generates the given event.

6. TCL SCRIPT EXAMPLE

The previous section provided details on the TCL support found in the Catalyst 6500, and how a TCL script can be loaded and registered with the system. We will now explore in more detail a sample TCL script providing some insight into the interaction between the script and the system.

The example script we shall explore is one that monitors the system for an interface down, and upon identifying this situation, will run a TDR diagnostics command on that interface, capture the show interface output, and send an e-mail to the administrator. The full script is shown in Figure 5.

Figure 5. Sample TCL Script

```
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90
#------------------------------------------------------------------
# EEM policy to monitor for a specified syslog message.
# Designed to be used for syslog interface-down messages.
# When event is triggered, a TDR check, Interface counters, and
# the optional given config commands will be run.
#
# January 2006 ISBU TME Team
#
# Copyright (c) 2006 by cisco Systems, Inc.
# All rights reserved.
#------------------------------------------------------------------
### The following EEM environment variables are used:
###
### _syslog_pattern (mandatory)        - A regular expression pattern match string
###                                      that is used to compare syslog messages
###                                      to determine when policy runs
### Example: _syslog_pattern            .*UPDOWN.*FastEthernet0/0.*
###
### _router_name (mandatory)           - A string used to indicate the name of
###                                      the device you are working on and emails
###                                      sent out will be tagged with this name
###                                      sevt-pod1
###
### _email_server (mandatory)          - A Simple Mail Transfer Protocol (SMTP)
###                                      mail server used to send e-mail.
### Example: _email_server               mailserver.customer.com
###
### _email_from (mandatory)            - The address from which e-mail is sent.
### Example: _email_from                 devtest@customer.com
###
### _email_to (mandatory)              - The address to which e-mail is sent.
### Example: _email_to                   engineering@customer.com
###
### _email_cc (optional)               - The address to which the e-mail must
###                                      be copied.
### Example: _email_cc                   manager@customer.com
###
### _config_cmd1 (optional)            - The first configuration command that
###                                      is executed.
### Example: _config_cmd1                interface Ethernet1/0
###
### _config_cmd2 (optional)            - The second configuration command that
###                                      is executed.
### Example: _config_cmd2                no shutdown
###
# check if all the env variables we need exist
# If any of them doesn't exist, print out an error msg and quit
if {![info exists _email_server]} {
set result \
"Policy cannot be run: variable _email_server has not been set"
```

```
error $result $errorInfo
}
if {![info exists _router_name]} {
set result \
"Policy cannot be run: variable _router_name has not been set"
error $result $errorInfo
}
if {![info exists _email_from]} {
set result \
"Policy cannot be run: variable _email_from has not been set"
error $result $errorInfo
}
if {![info exists _email_to]} {
set result \
"Policy cannot be run: variable _email_to has not been set"
error $result $errorInfo
}
if {![info exists _email_cc]} {
#_email_cc is an option, must set to empty string if not set.
set _email_cc ""
}
namespace import ::cisco::eem::*
namespace import ::cisco::lib::*
# 1. query the information of latest triggered eem event
array set arr_einfo [event_reqinfo]
if {$_cerrno != 0} {
set result [format "component=%s; subsys err=%s; posix err=%s;\n%s" \
$_cerr_sub_num $_cerr_sub_err $_cerr_posix_err $_cerr_str]
error $result
}
set msg $arr_einfo(msg)
set config_cmds ""
if {[regexp {.*Interface (.*),.*} $msg \
match intf_match]} {
}
# 2. execute the user-defined config commands
if [catch {cli_open} result] {
error $result $errorInfo
} else {
array set cli1 $result
}
if [catch {cli_exec $cli1(fd) "en"} result] {
error $result $errorInfo
}
if [catch {cli_exec $cli1(fd) "test cable-diagnostics tdr interface $intf_match"} result] {
error $result $errorInfo
}
if [catch {cli_exec $cli1(fd) "show interface $intf_match counters errors"} result] {
error $result $errorInfo
} else {
set cmd1_output $result
}
after 5000
if [catch {cli_exec $cli1(fd) "show cable-diagnostics tdr interface $intf_match"} result] {
error $result $errorInfo
} else {
set cmd2_output $result
}
if [catch {cli_exec $cli1(fd) "config t"} result] {
error $result $errorInfo
}
if {[info exists _config_cmd1]} {
if [catch {cli_exec $cli1(fd) $_config_cmd1} result] {
error $result $errorInfo
}
append config_cmds $_config_cmd1
```

```
}
if {[info exists _config_cmd2]} {
if [catch {cli_exec $cli1(fd) $_config_cmd2} result] {
error $result $errorInfo
}
append config_cmds "\n"
append config_cmds $_config_cmd2
}
if [catch {cli_exec $cli1(fd) "end"} result] {
error $result $errorInfo
}
if [catch {cli_close $cli1(fd) $cli1(tty_id)} result] {
error $result $errorInfo
}
after 1000
# 3. send the notification email
if [catch {smtp_subst $intchk_template} result] {
error $result $errorInfo
}
if [catch {smtp_send_email $result} result] {
error $result $errorInfo
}
# open a cli connection
if [catch {cli_open} result] {
error $result $errorInfo
} else {
array set cli $result;  }
```

The first line of the script includes the mandatory EEM configuration line as follows:

```
::cisco::eem::event_register_syslog occurs 1 pattern $_syslog_pattern maxrun 90
```

This line identifies the script will use the SYSLOG Event Detector to parse SYSLOG messages for a pattern match that is defined by the value set in the environment variable "_syslog_pattern." A quick look at the CLI shows that the administrator has set the environment variable as "LINK-3-UPDOWN.*state to down" as shown in the following output.

```
C6500#show event manager environment _syslog_pattern
LINK-3-UPDOWN.*state to down
```

Following the first line is a series of comment lines (line 2 through to line 48), which describe the purpose of the script, date of script, author, copywrite information, and environment variables that are used in the script. While not mandatory, it is good documentation practice to document information relevant to this script at the beginning.

Lines 49 through to 68 check that each of the required environment variables has been set by the administrator prior to executing any subsequent TCL statements. If any one of the environment variables has not been set then the script will print out an error message on the console and cease to run. A quick check from the switch CLI can also confirm that each of the environment variables has been set. This is shown in the following CLI output.

```
C6500#show event manager environment all
No.  Name                          Value
1    _email_server                 10.200.1.1
2    _email_from                   eem@cisco.com
3    _email_to                     admin@abcxyz.com
4    _router_name                  C6500
```

All of the mandatory environment variables have been set, so the script will continue to execute subsequent statements.

The “namespace” commands at lines 69 and 70 are used to import environment extensions that can be used by policy developers. The EEM namespace imports the commands in Table 2.

Table 2. EEM Namespace Imported Keywords

Category	Description
EEM Event Registration Keywords	Adds support for event_register keywords such as event_register_cli, event_register_counter, event_register_interface, event_register_snmp, and many more
EEM Event Information Keywords	Adds support for event_reqinfo
EEM Event Publish Keywords	Adds support for the event_publish keyword
EEM Action Keywords	Adds keywords like action_reload, action_script, action_snmp_trap, action_syslog, and many more
EEM Utility Keywords	Support for utility keywords like timer_arm, timer_cancel, register_counter, and many more
EEM Context Library	Support for context_save and context_retrieve
EEM System Information Keywords	Support for sys_reqinfo_snmp, sys_reqinfo_routername, and many more
CLI Library Keywords	Adds cli_open, cli_exec, cli_read, cli_write, and many more

The second namespace that is imported is the cisco::lib namespace, which adds support for SMTP related commands such as smtp_send_email. Should any of these commands be required to be executed in your TCL script, the associated namespace must be imported in the script.

Statements following the above are directly related to the actions that the policy invokes on the generation of the event. Lines 71 through 81 query the triggered event and set local variables with information related to that event. The reference to “Cerrno” relates to a Cisco error number generated by the switch when the event occurs. Whenever the “_cerrno” variable is set, four other TCL global variables, namely “_cerr_sub_num,” “_cerr_sub_err,” “_cerr_posix_err,” and “_cerr_str,” are derived from “_cerrno” and are also set.

Lines 82 through to 98 invoke a set of commands to run on the switch. To do this, the script opens access to the CLI and sends three commands to it, namely those in the list below.

1. en (shortcut for enable) to gain access to the configuration mode
2. test cable-diagnostics tdr interface $intf_match (variable matches the interface that went down)
3. show interface $intf_match counters errors

Following the execution of these commands, the script waits 5 seconds (indicated by the command on line 99) and executes a command to show the results of the TDR command run (indicated in line 100).

Subsequent to this, the script offers two additional commands to be executed should the respective environment variables be set. Lines 105 through to 121 reference two environment variables (config_cmd1 and config_cmd2) to see if they have been set. If so, then they too are executed from the CLI. After these two commands have been interrogated for execution, access to the CLI is then closed for this script.

The script then waits one second (indicated by line 125) and then sends an e-mail alert using the commands from lines 126 through to 138. The e-mail is sent to the e-mail address set in the environment variable “_email_to” via the e-mail server indicated in the environment variable “_email_server.”

The output of this script is shown below. On the CLI, when an interface is shutdown, the following is an example of what will be seen.

```
1d17h: %LINEPROTO-5-UPDOWN: Line protocol on Interface GigabitEthernet2/1,
changed state to down

1d17h: %LINK-3-UPDOWN: Interface GigabitEthernet2/1, changed state to down
```

Note that the second message contains the “Link-3-Updown” message that the event detector is looking for. The script is kicked into action, and the resulting e-mail that is sent contains the output from the TDR run. An example of what the e-mail looks like is shown below.

```
From: eem@cisco.com
To: techo@cisco.com
Subject: Interface Gigabit 2/1 has gone down on C6500
Port        Align-Err    FCS-Err   Xmit-Err    Rcv-Err UnderSize OutDiscards
Gi2/1               0          0          0          0         0           0
Port      Single-Col Multi-Col  Late-Col Excess-Col Carri-Sen     Runts    Giants
Gi2/1              0         0         0          0         0         0         0
Port       SQETest-Err Deferred-Tx IntMacTx-Err IntMacRx-Err Symbol-Err
Gi2/1                0           0            0            0          3
C6500#
TDR test last run on: January 12 11:22:46
Interface Speed Pair Cable length        Distance to fault   Channel Pair status
--------- ----- ---- ------------------- ------------------- ------- ------------
Gi2/1     auto  1-2  N/A                 N/A                 Invalid Terminated
                3-4  N/A                 N/A                 Invalid Terminated
                5-6  N/A                 N/A                 Invalid Terminated
                7-8  N/A                 N/A                 Invalid Terminated
```

7. SUMMARY

Embedded Event Manager in the Catalyst 6500 provides network administrators with a flexible framework upon which to significantly enhance the operational manageability of those switches. The programmable nature of this policy-based framework allows customers to apply unique policy rules to meet local management requirements while at the same time ensuring that those policies can be enacted in a time saving manner. This can help to provide the administration team with more real time information that can lead to better problem identification and resolution.

CISCO SYSTEMS

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.com 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

Printed in USA C11-366432-00 09/06