EDCS-984017

Cisco 5940 Series Embedded Services Router

Common Criteria

Operational User Guidance

and

Preparative Procedures

Version 0.3

May 2011

Table of Contents

DOCUMENT INTRODUCTION

This document provides supporting evidence for an evaluation of a specific Target of Evaluation (ToE), the Cisco 5940 Series Embedded Services Router (ESR). This Operational User Guidance with Preparative Procedures addresses the administration of the ToE software and hardware and describes how to install, configure, and maintain the ToE in the Common Criteria evaluated configuration.

1. Introduction

This Operational User Guidance with Preparative Procedures documents the administration of the Cisco 5940 Series Embedded Services Router (ESR) ToE certified under Common Criteria. The Cisco 5940 Series Embedded Services Router (ESR) ToE may be referenced below as the 5940, ESR, TOE, or simply router.

1.1 Audience

This document is written for administrators configuring the ToE. This document assumes that you are familiar with the basic concepts and terminologies used in internetworking, and understand your network topology and the protocols that the devices in your network can use, that you are a trusted individual, and that you are trained to use the operating systems on which you are running your network.

1.2 Purpose

This document is the Operational User Guidance with Preparative Procedures for the Common Criteria evaluation. It was written to highlight the specific ToE configuration and administrator functions and interfaces that are necessary to configure and maintain the ToE in the evaluated configuration. This document is not meant to detail specific actions performed by the administrator but rather is a road map for identifying the appropriate locations within Cisco documentation to get the specific details for configuring and maintaining 5940 ESR operations.

1.3 Document References

This document makes reference to several Cisco Systems documents. The documents used are shown below.

[1]	Release Notes for Cisco IOS Release 15.1(2)GC, January, 31, 2011	http://www.cisco.com/en/US/partner/docs/solutions/GGSG-Engineering/15_1_2GC/Release_Notes/RN_15_1_2GC.html
[2]	Cisco 5940 Embedded Services Router Hardware Technical Reference Guide, Last Updated: January 25, 2011	http://www.cisco.com/en/US/docs/solutions/GGSG-Engineering/Dusty/hardware/guide/5940hw.pdf
[3]	Cisco IOS Configuration Fundamentals Configuration Guide, Release 15.1	http://www.cisco.com/en/US/partner/docs/ios/fundamentals/configuration/guide/15_1/cf_15_1_Book.html
[4]	Cisco IOS Security Configuration Guide: Securing User Services, Cisco IOS Release 15.1M&T	sec_user_services_15_1_book.pdf or http://www.cisco.com/en/US/docs/ios/sec_user_services/configuration/guide/15_1/sec_user_services_15_1_book.html

[5]	Network Management Configuration Guide, Cisco IOS Release 15.1M&T	http://www.cisco.com/en/US/partner/docs/ios/netmgmt/configuration/guide/15_1/nm_15_1_book.html
[6]	Software Configuration Guide for Cisco IOS Release 15.1(2)GC	http://www.cisco.com/en/US/docs/solutions/GGSG-Engineering/Beetlejuice/Configuration/15_1_2GC.pdf
[7]	Cisco IOS Security Command Reference	http://www.cisco.com/en/US/docs/ios/security/command/reference/sec_cr_book.pdf

1.4 Supported Hardware and Software

Only the following hardware and software listed below is compliant with the Common Criteria EAL2 evaluation. Using hardware not specified invalidates the secure configuration. Likewise, using any software version other than the evaluated software listed below will invalidate the secure configuration.

1.4.1 Supported Configurations

- Cisco 5940 ESR Conduction-cooled cPCI router module running IOS 15.1(2)GC1

  or

- Cisco 5940 ESR Air-cooled cPCI router module running IOS 15.1(2)GC1

1.5 Operational Environment

1.5.1 Required software for the operational environment

Component	Usage/Purpose Description for TOE performance
Router Chassis	The host chassis for the router provides the power to the module as well as directly connected Ethernet interfaces via a standard cPCI backplane. The chassis selected can be any off –the-shelf chassis that supports 3U cPCI cards.
Rear Transition Module (RTM)	The RTM supports Input/Ouput connectors through standard RJ-45 connectors, or any other cPCI compatible network connector. The RTM can be any off-the-shelf module that is a cPCI form factor. It is installed in the chassis directly opposite the card with which it is to be used.
Management Workstation with SSH Client	This includes any IT Environment Management workstation with a SSH client installed that is used by the TOE administrator to support TOE administration through SSH protected channels. Any SSH client that supports SSH v2 may be used.
AAA Server	This includes any IT environment AAA server that provides single-use authentication mechanisms. This can be any AAA server that provides single-use authentication. The TOE correctly leverages the services provided by this AAA server to provide single-use authentication to administrators.

Component	Usage/Purpose Description for TOE performance
Syslog Server	A syslog server with the capability to support TLS-protected TCP syslog communications is required for use with the TOE.
Certificate Authority (CA)	This includes any IT Environment Certificate Authority on the TOE network. This can be used to provide the TOE with a valid certificate during certificate enrollment.

1.5.2 Optional software for the operational environment:

Component	Usage/Purpose Description for TOE performance
VPN Peer	This includes any peer with which the TOE participates in VPN communications. VPN peers may be any device or software client that supports IPSec communications. Both VPN clients and VPN gateways are considered VPN peers by the TOE.
NTP Server	The TOE supports communications with an NTP server. A solution must be used that supports MD5 hashing of communications with up to a 32 character key.

1.6 Excluded Functionality

Excluded Functionality	Exclusion Rationale
Non-FIPS 140-2 mode of operation on the router.	This mode of operation includes non-FIPS allowed operations.
Telnet for management purposes.	Telnet passes authentication credentials in clear text. SSHv2 is to be used instead.

2. Secure Acceptance of the TOE

In order to ensure the correct TOE is received, the TOE should be examined to ensure that that is has not been tampered with during delivery.

Verify that the TOE software and hardware were not tampered with during delivery by performing the following actions:

Step 1 Before unpacking the TOE, inspect the physical packaging the equipment was delivered in. Verify that the external cardboard packing is printed with the Cisco Systems logo and motifs. If it is not, contact the supplier of the equipment (Cisco Systems or an authorized Cisco distributor/partner).

Step 2 Verify that the packaging has not obviously been opened and resealed by examining the tape that seals the package. If the package appears to have been resealed, contact the supplier of the equipment (Cisco Systems or an authorized Cisco distributor/partner).

Step 3 Verify that the box has a white tamper-resistant, tamper-evident Cisco Systems bar coded label applied to the external cardboard box. If it does not, contact the supplier of the equipment (Cisco Systems or an authorized Cisco distributor/partner). This label will include the Cisco product number, serial number, and other information regarding the contents of the box.

Step 4 Note the serial number of the TOE on the shipping documentation. The serial number displayed on the white label affixed to the outer box will be that of the device. Verify the serial number on the shipping documentation matches the serial number on the separately mailed invoice for the equipment. If it does not, contact the supplier of the equipment (Cisco Systems or an authorized Cisco distributor/partner).

Step 5 Verify that the box was indeed shipped from the expected supplier of the equipment (Cisco Systems or an authorized Cisco distributor/partner). This can be done by verifying with the supplier that they shipped the box with the courier company that delivered the box and that the consignment note number for the shipment matches that used on the delivery. Also verify that the serial numbers of the items shipped match the serial numbers of the items delivered. This verification should be performed by some mechanism that was not involved in the actual equipment delivery, for example, phone/FAX or other online tracking service.

Step 6 Once the TOE is unpacked, inspect the unit. Verify that the serial number displayed on the unit itself matches the serial number on the shipping documentation and the invoice. If it does not, contact the supplier of the equipment (Cisco Systems or an authorized Cisco distributor/partner). Also verify that the unit has the following external identification:

TOE Model	External Identification
Conduction Cooled card	The part number CISCO5940RC-K9 on the Serial Number tag below the serial

	number (as in the image below)
Air Cooled card	The part number CISCO5940RA-K9 on the Serial Number tag below the serial number (similar to the image below)

Step 7 Approved methods for obtaining a Common Criteria evaluated software images:

- Download the Common Criteria evaluated software image file from Cisco.com onto a trusted computer system. Software images are available from Cisco.com at the following: http://www.cisco.com/cisco/web/download/index.html.
- The TOE ships with the correct software images installed.

Step 8 Once the file is downloaded, verify that it was not tampered with by using an MD5 utility to compute an MD5 hash for the downloaded file and comparing this with the MD5 hash for the image listed in Table 5 below. If the MD5 hashes do not match, contact Cisco Technical Assistance Center (TAC) http://tools.cisco.com/ServiceRequestTool/create/launch.do.

Step 9 Install the downloaded and verified software image onto your 5940 ESR as described in [**3**] Loading and Managing System Images → Working with System Images

Start your 5940 ESR as described in [**3**]. Confirm that your 5940 ESR loads the image correctly, completes internal self-checks and displays the cryptographic export warning on the console.

Step 10 The end-user must confirm once the ToE has booted that they are indeed running the evaluated version. Use the "**show image**" command to display the currently running system image filename and the system software release version.

Table 1: Evaluated Software Images

Software Version	**Image Name**	**MD5 hash**
IOS 15.1(2)GC1	c5940-adventerprisek9-mz.SPA.151-	66a47d02717279448f720d2d1d6e3538

	2.GC1.bin	

3. Secure Installation and Configuration

3.1 Physical Installation

Follow the Cisco 5940 Embedded Services Router Hardware Technical Reference Guide [2] for preparation of the physical site, and hardware installation. There are separate sections "Installing the Cisco 5940 ESR Air-Cooled Card" and "Installing the Cisco 5940 ESR Conduction-Cooled Card".

Note that the Rear Transition Module (RTM) should also be installed in the chassis at this time by following the instructions in [**2**].

3.2 Initial Setup via Direct Console Connection

The ESRs must be given basic configuration via console connection prior to being connected to any network.

3.2.1 Options to be chosen during the initial setup of the ESR 5940

When you run the "setup" command, or after initially turning on the router you are free to choose answers that fit your policies with the exception of the following values.

1 – Enable Secret – Must adhere to the password complexity requirements. Note that this setting can be confirmed after "setup" is complete by examining the configuration file for "enable secret 5 …"

2 – Enable Password - Must adhere to the password complexity requirements. Note that this must be set to something different than the enable secret during "setup", however after setup this will not be used within the evaluated configuration.

3 – Virtual Terminal Password - Must adhere to the password complexity requirements. Note that securing the virtual terminal (or vty) lines with a password in the evaluated configuration is suggested. This password allows access to the device through only the console port. Later in this guide steps will be given to allow ssh into the vty lines.

4 – Configure SNMP Network Management – NO (this is the default). Note that this setting can be confirmed after "setup" is complete by examining the configuration file to ensure that there is no "snmp-server" entry.

3.2.2 Saving Configuration

IOS uses both a running configuration and a starting configuration. Configuration changes affect the running configuration, in order to save that configuration the running configuration (held in memory) must be copied to the startup configuration. This may be achieved by either using the **write memory** command or the **copy system:running-config nvram:startup-config** command. These commands should be used frequently when making changes to the configuration

of the Router. If the Router reboots and resumes operation when uncommitted changes have been made, these changes will be lost and the Router will revert to the last configuration saved.

3.2.3 Enabling FIPS Mode

The TOE must be run in the FIPS mode of operation. This is done by setting the following in the configuration:

The value of the boot field must be 0x0102. This setting disables break from the console to the ROM monitor and automatically boots the IOS image. From the ROMMON command line enter the following:

config-register 0x0102

3.2.4 Access Control and Lockout

The ESR must be configured to use a username and password for each administrator and one password for the enable command. Ensure all passwords are stored encrypted by using the following command:

service password-encryption

When creating administrator accounts, all individual accounts are to be set to a privilege level of one. This is done by using the following commands:

username <name> password <password>

to create a new username and password combination, and

username <name> privilege 1

to set the privilege level of <name> to 1.

Administrator account access is to be restricted to a specified number of authentication attempts before the administrator account in question is locked out. The account then requires unlocking by an authorized administrator before it can be used again. The evaluated configuration requires that the lockout occurs after a specified threshold for unsuccessful authentication attempts. Use the following command, with '<x>' being the required number of attempts before lockout, to set the authentication failure threshold (the authentication threshold must be non zero):

aaa local authentication attempts max-fail <x>

A locked user account may be unlocked by a privileged administrator by using the following command:

clear aaa local user lockout <username>

3.3 Network Protocols and Cryptographic Settings

3.3.1 Remote Administration Protocols

- Telnet for management purposes is disabled by default and must remain disabled in the evaluated configuration.
- SSHv2 must be used. To enable sshv2, use the "**ip ssh version 2**" command.
- HTTP and HTTPS servers were not evaluated and must be disabled:
  no ip http server
  no ip http secure-server
- SNMP server was not evaluated and must be disabled: **no snmp-server**

3.3.2 Authentication Server Protocols

- RADIUS (outbound) for authentication of ToE administrators to remote authentication servers are disabled by default but should be enabled by administrators in the evaluated configuration.
  - To configure RADIUS refer to **[4]** Authentication, Authorization, and Accounting (AAA) → Authentication → Configuring Authentication → How to Configure AAA Authentication Methods → Configuring Login Authentication Using AAA → Login Authentication Using Group RADIUS. Use best practice for selection and protection of a key to ensure that the key is not easily guessable and is not shared with unauthorized users.

3.3.3 Logging Configuration

- Logging of command execution must be enabled:

  Router(config)#**archive**

  Router(config-archive)#**log config**

  Router(config-archive-log-cfg)#**logging enable**

  Router(config-archive-log-cfg)#**hidekeys**

  Router(config-archive-log-cfg)#**notify syslog**

  Router(config-archive-log-cfg)#**exit**

  Router(config-archive)#**exit**
- Add year to the timestamp:
  Router(config)# **service timestamps log datetime year**
- Enable radius and ssh debugging:
  debug radius authentication

- Set the size of the logging buffer. It is recommended to set it to at least 150000000:
  logging buffer 150000000
- To generate logging messages for failed and successful login attempts in the evaluated configuration, issue the login on-failure and login on-success commands:
  Router(config)#login on-failure log
  Router(config)#login on-success log
- Syslog (outbound) for transmission of syslog events to a remote syslog server is disabled by default but must be enabled in the evaluated configuration. SSL-protected syslog is to be used.
  - To configure syslog, refer to the instructions below in 'Logging Fail-Close'
- Timestamps must be enabled for the audit records on the ESR: **service timestamps log datetime**

3.3.4 Logging Fail-Close

To protect against audit data loss the TOE requires that records be sent off the router to an external TCP syslog server. In the event that the external server cannot be reached by the router new traffic sessions through the router will be stopped, and an alert event will be logged to alert the privileged administrators. New VPN sessions will also be denied. The router will continue to attempt to connect to the external server, and once a connection is re-established new connections will resume.

Since this functionality is not enabled by default the following command must be entered at the CLI to configure this option: **logging host <ip> transport tcp port <port> audit**. The following sets the TOE to use a host at 10.150.0.206 that is listening on tcp port 1470.

Router# **configure terminal**

Router(config)#**logging host *10.150.0.206* transport tcp port *1470* audit**

Router(config)#**exit**

This function allows for the firewall to disallow any new connection attempts in the audit logging system failure.

3.3.5 Logging Protection

Protection must be provided for the syslog server communication. If the syslog server is not directly co-located with the TOE, then the syslog server must be located in a physically protected facility and connected to a router capable of establishing an IPSec tunnel with the TOE. This will protect the syslog records as they traverse the public network.

Following are sample instructions to configure the TOE to support an IPSec tunnel with aes encryption, with 11.1.1.4 as the peer router, 10.1.1.7 and 11.1.1.6 as the local IPs, and the syslog server on the 12.1.1.0 /28 subnet:

5940-common-criteria#**configure terminal**
Enter configuration commands, one per line. End with CNTL/Z.
5940-common-criteria(config)#**crypto isakmp policy 1**
5940-common-criteria(config-isakmp)#**encryption aes**
5940-common-criteria(config)#**crypto ipsec transform-set rtpset esp-aes esp-sha-hmac**
5940-common-criteria(config)#**crypto map rtp 1 ipsec-isakmp**
5940-common-criteria(config-crypto-map)#**set peer 11.1.1.4**
5940-common-criteria(config-crypto-map)#**set transform-set rtpset**
5940-common-criteria(config-crypto-map)#**match address 115**
5940-common-criteria(config-crypto-map)#**exit**
5940-common-criteria(config)#**interface g0/1**
5940-common-criteria(config-if)#**ip address 10.1.1.7 255.255.255.0**
5940-common-criteria(config-if)#**no ip route-cache**
5940-common-criteria(config-if)#**crypto map rtp**
5940-common-criteria(config-if)#**interface g0/0**
5940-common-criteria(config-if)#**ip address 11.1.1.6 255.255.255.0**
5940-common-criteria(config-if)#**crypto map rtp**
5940-common-criteria(config-if)#**exit**
5940-common-criteria(config)#**ip route 12.1.1.0 255.255.255.0 11.1.1.4**
5940-common-criteria(config)#**access-list 115 permit ip 10.1.1.0 0.0.0.255 12.1.1.0 0.0.0.255 log**

3.3.6 Base Firewall Rule set Configuration

The evaluated configuration requires that the TOE Access control lists (ACLs) are to be configured to drop all packet flows as the default rule. This can be achieved by including an ACL rule to drop all packets as the last rule in the ACL configuration.

A privileged authorized administrator may manipulate the ACLs using the commands ip inspect, access-list and access-group as described **[7]**.

Access lists must be configured on the TOE to prevent spoofing of external or internal addresses through the opposite interface and also to block broadcast source address and loopback source address traffic. These access control lists are required for compliance with the Application Firewall Protection Profile.

Note: These access lists must be integrated with the defined security policy for your TOE router. Enabling just these access lists with no permits will result in traffic being dropped.

In this example, we are assuming that interface GigabitEthernet0/0 is the external interface, and is assigned an IP address of 10.200.1.1. Interface

GigabitEthernet0/1 is the internal interface and is assigned an IP address of 10.100.1.1.

To prevent the passing of traffic with an internal source address on the external Interface, apply the following access control list to the external interface:

Router# **configure terminal**

Router(config)# **access-list 199 deny ip 10.100.0.0 0.0.255.255 any log-input**

To prevent the passing of traffic with an external source address on the internal Interface, apply the following access control list to the internal interface:

Router# **configure terminal**

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)# **access-list 100 deny ip 10.200.0.0 0.0.255.255 any log-input**

To prevent the passing of traffic with a broadcast or loopback address on either interface, apply the following access control list to both interfaces:

Router# **configure terminal**

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)# **access-list 100 deny ip 224.0.0.0 15.255.255.255 any log-input**

Router(config)# **access-list 100 deny ip 255.255.255.255 0.0.0.0 any log-input**

Router(config)# **access-list 100 deny ip 127.0.0.0 0.255.255.255 any log-input**

Router(config)# **access-list 199 deny ip 224.0.0.0 15.255.255.255 any log-input**

Router(config)# **access-list 199 deny ip 255.255.255.255 0.0.0.0 any log-input**

Router(config)# **access-list 199 deny ip 127.0.0.0 0.255.255.255 any log-input**

If remote administration is required, ssh has to be explicitly allowed through either the internal or external interfaces.

Router# **configure terminal**

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)# **access-list 199 permit tcp host 10.200.0.1 host 10.200.0.1 eq 22 log-input**

To close ports that don't need to be open and may introduce additional vulnerabilities, implement the following acl:.

Router(config)# **access-list 100 deny 132 any any log-input**

Router(config)# **access-list 199 deny 132 any any log-input**

To apply the acls to the interfaces:

Router(config)# **interface GigabitEthernet0/0**

Router(config-if)# **ip access-group 199 in**

Router(config)# **interface GigabitEthernet0/1**

Router(config-if)# **ip access-group 100 in**

4. Secure Management

4.1 User Roles

The ESR has both privileged and non-privileged administrator roles. These roles are configured in the Access Control and Lockout section above. Non-privileged roles are those that have authenticated but not entered the enable password. Privileged roles are those that have authenticated and then entered the enable password.

4.2 Passwords

To prevent users from choosing insecure passwords, each password must meet the following requirements:

- At least eight characters long
- Does not contain more than three consecutive characters, such as abcd
- Does not contain more than two repeating characters, such as aaabbb
- Does not contain dictionary words
- Does not contain common proper names

This requirement applies to the local password database and on the password selection functions provided by the ToE, but remote authentication servers may have pre-configured passwords which do not meet the quality metrics.

4.3 Clock Management

Clock management is restricted to the privileged administrator.

For instructions to set the clock, refer to "Basic System Management" → "Performing Basic System Management" → "Setting Time and Calendar Services" **[5]**.

This section contains information on setting the local hardware clock or NTP sources.

4.4 Identification and Authentication

Configuration of Identification and Authentication settings is restricted to the privileged administrator.

The ESR can be configured to use any of the following authentication methods:

- Remote authentication (RADIUS)
  - Refer to "Authentication Server Protocols" elsewhere in this document for more details.
- Local authentication (password or SSH public key authentication);
  - Note, this should only be configured for local fallback if the remote authentication server is not available.

4.5 Information Flow Policies

The TOE may be configured for information flow control/ firewall rules. Guidance for configuration of IOS Intrusion Prevention/ Intrusion Detection is located at:

IPv4:
http://www.cisco.com/en/US/partner/docs/ios/sec_data_plane/configuration/guide/sec_zone_polcy_firew.html

IPv6: http://www.cisco.com/en/US/partner/docs/ios/ipv6/configuration/guide/ip6-sec_trfltr_fw_ps10592_TSD_Products_Configuration_Guide_Chapter.html

Configuration of information flow policies is restricted to the privileged administrator.

4.6 Intrusion Prevention System (IPS)

The TOE may be configured for Intrusion Prevention capabilities. Guidance for configuration of IOS Intrusion Prevention/ Intrusion Detection is located at:

http://www.cisco.com/en/US/partner/docs/ios/sec_data_plane/configuration/guide/sec_cfg_ips_ps6441_TSD_Products_Configuration_Guide_Chapter.html

Configuration of IPS settings is restricted to the privileged administrator.

4.7 Virtual Private Networks (VPN)

6.2.2.1. Internet Key Exchange Configuration

The TOE all allows the Privilged Administrator to configure Internet Key Exchange (IKE) policies. IKE is a key management protocol standard that is used in conjunction with the IP Security (IPSec) standard. IPSec is a feature that provides robust authentication and encryption of IP packets.

IKE is a hybrid protocol that implements the Oakley key exchange and the Skeme key exchange inside the Internet Security Association and Key Management Protocol (ISAKMP) framework. (ISAKMP, Oakley, and Skeme are security protocols implemented by IKE.) IPSec can be configured without IKE, but IKE enhances IPSec by providing additional features, flexibility, and ease of configuration for the IPSec standard.

Additional information regarding Configuration of IKE Policies can be found in Internet Key Exchange Security Protocol Commands of [7].

- This functionality is available to the Privileged Administrator.

6.2.2.2. IPSec Configuration

IPSec provides security for transmission of sensitive information over unprotected networks such as the Internet. IPSec acts at the network layer, protecting and authenticating IP packets between participating IPSec devices (“peers”), such as Cisco routers.

With IPSec, data can be sent across a public network without observation, modification, or spoofing, which enables applications, such as Virtual Private Networks (VPNs).

Additional information regarding configuration of IPSec can be found in the IPSec Network Security Commands of [7].

• This functionality is available to the Privileged Administrator.Configuration of VPN settings is restricted to the privileged administrator.

4.8 Certificate Server

Certificates can be used for IKE authentication. To identify the trustpoint that is used to validate a certificate during Internet Key Exchange (IKE) authentication, use the **ca trust-point** command in ISAKMP profile configuration mode. For instructions on the usage of this command see the "ca trust-point" command in **[7]**.

5. Security Relevant Events

ESR maintains logs in multiple locations: local storage of the generated audit records, and simultaneous offload of those events to the external syslog server. For the most complete view audited events, across all devices, and to view the auditable events defined in the Security Target administrators should review the Audit Log.

5.1 Reviewing, Sorting, and Filtering Audited Events

Using the ESR Command Line Interface (CLI) administrators can review, sort and filter audited events based on presumed subject address, ranges of dates, ranges of times, and ranges of addresses.

- To review locally stored audit records enter the command "**show logging**" To perform searching/sorting, see the instructions in Annex A: The Search/Sort Script.

5.2 Deleting Audit Records

The TOE provides the privileged Administrator the ability to delete audit records audit records stored within the TOE.

This is done with the **clear logging** command.

5.3 Administering Logging Fail-Close

To protect against audit data loss the TOE requires that records be sent off the router to an external TLS-capable TCP syslog server. In the event that the external

server cannot be reached by the router new traffic sessions through the router will be stopped, and an alert event will be logged to alert the privileged administrators. New VPN sessions will also be denied. The router will continue to attempt to connect to the external server, and once a connection is re-established new connections will resume. See Logging Fail-Close above for configuration details.

When the ESR loses the connection to the syslog server, sometimes it will re-establishes the TCP syslog session on its own and other times it won't do so automatically.

The router will attempt to reconnect for a limited time. The limited reconnect retries may be exhausted by the time the syslog server comes back operational or when the IP connection stabilizes, and hence the syslog server may appear not connected.

The workaround for this issue is to remove the "logging host ..." entry for that syslog server and re-add it. By adding it back via the CLI the connection retry is reset to the maximum and the syslog server should reconnect.

6. Modes of Operation

An IOS router has several modes of operation, these modes are as follows:

Booting – while booting, the routers drop all network traffic until the router image and configuration has loaded. This mode of operation automatically progresses to the Normal mode of operation. During booting, a user may press the break key on a console connection within the first 60 seconds of startup to enter the ROM Monitor mode of operation. This Booting mode is referred to in the IOS guidance documentation as "ROM Monitor Initialization". Additionally if the Router does not find a valid operating system image it will enter ROM Monitor mode and not normal mode therefore protecting the router from booting into an insecure state.

Normal - The IOS router image and configuration is loaded and the router is operating as configured. It should be noted that all levels of administrative access occur in this mode and that all router based security functions are operating. While operating the router have little interaction with the administrator. However, the configuration of the router can have a detrimental effect on security. Misconfiguration of the router could result in the unprotected network having access to the internal/protected network

ROM Monitor – This mode of operation is a maintenance, debugging and disaster recovery mode. While the router is in this mode, no network traffic is routed between the network interfaces. In this state the router may be configured to upload a new boot image from a specified TFTP server, perform configuration tasks and run various debugging commands. It should be noted that while no administrator password is required to enter ROM monitor mode, physical access to the router is required, therefore the router should be stored in a physically secure location to avoid unauthorized access which may lead to the router being placed in an insecure state.

Following operational error the router reboots (once power supply is available) and enters booting mode.

7. Security Measures for the Operational Environment

Proper operation of the TOE requires functionality from the environment. It is the responsibility of the authorized users of the TOE to ensure that the TOE environment provides the necessary functions, and adheres to the assumptions listed below. The assumption identifiers map to the assumptions as defined in the Security Target.

Environment Security Objective	IT Environment Security Objective Definition	Privileged and Non-privileged administrator responsibility
A.PHYSEC	The TOE is physically secure.	Maintain the physical protection of the TOE in a manner consistent with the implementing organization's security policies.
A.LOWEXP	The threat of malicious attacks aimed at discovering exploitable vulnerabilities is considered low.	No required action.
A.GENPUR	There are no general-purpose computing capabilities (e.g., the ability to execute arbitrary code or applications) and storage repository capabilities on the TOE.	Use the machine as intended for routing, VPN or IPS functionality.
A.PUBLIC	The TOE does not host public data.	Use the machine as intended for routing, VPN or IPS functionality.
A.NOEVIL	Authorized administrators are non-hostile and follow all administrator guidance; however, they are capable of error.	Users must be properly trained in the usage and proper operation of the TOE and all the provided functionality per the implementing organization's operational security policies. These users must follow the provided guidance.
A.SINGEN	Information can not flow among the internal and external networks unless it passes through the TOE.	Ensure that the TOE is appropriately placed in the network at the required locations to enforce the organization's operational security policies.
A.DIRECT	Human users within the physically secure boundary protecting the TOE may attempt to access the TOE from some direct connection (e.g., a console port) if the connection is part of the TOE.	Protect the physical console port on the RTM from unauthorized access.
A.NOREMO	Human users who are not authorized administrators can not access the TOE remotely from the internal or external networks.	Follow the authentication information in this document.

Environment Security Objective	IT Environment Security Objective Definition	Privileged and Non-privileged administrator responsibility
A.REMACC	Authorized administrators may access the TOE remotely from the internal and external networks.	Follow the authentication information in this document.
OE.GUIDAN	The TOE must be delivered, installed, administered, and operated in a manner that maintains security.	Users must be properly trained in the usage and proper operation of the TOE and all the provided functionality per the implementing organization's operational security policies. These users must follow the provided guidance.
OE.ADMTRA	Authorized administrators are trained as to establishment and maintenance of security policies and practices.	Users must be properly trained in the usage and proper operation of the TOE and all the provided functionality per the implementing organization's operational security policies. These users must follow the provided guidance.
OE.NTP	The IT environment may be configured with an NTP server that is able to provide reliable time to the TOE. The communications must be protected using MD5 hashing with up to a 32 character key.	Configure the NTP server as highlighted in this row.
OE.SYSLOG	The IT environment must supply a syslog server capable of receiving TLS-protected TCP syslog information.	Configure the TOE for TLS-protected syslog.

8. Related Documentation

Use this document in conjunction with the IOS 15.1 documentation at the following location:

- http://www.cisco.com/

Obtaining Documentation

The following sections provide sources for obtaining documentation from Cisco Systems.

8.1 World Wide Web

You can access the most current Cisco documentation on the World Wide Web at the following sites:

- http://www.cisco.com
- http://www-china.cisco.com
- http://www-europe.cisco.com

8.2 Ordering Documentation

Cisco documentation is available in the following ways:

Registered Cisco Direct Customers can order Cisco Product documentation from the Networking Products MarketPlace:

http://www.cisco.com/cgi-bin/order/order_root.pl

Registered Cisco.com users can order the Documentation CD-ROM through the online Subscription Store:

http://www.cisco.com/go/subscription

Non-registered Cisco.com users can order documentation through a local account representative by calling Cisco corporate headquarters (California, USA) at 408 526-7208 or, in North America, by calling 800 553-NETS (6387).

8.3 Documentation Feedback

If you are reading Cisco product documentation on the World Wide Web, you can submit technical comments electronically. Click Feedback in the toolbar and select Documentation. After you complete the form, click Submit to send it to Cisco.

You can e-mail your comments to bug-doc@cisco.com.

To submit your comments by mail, for your convenience many documents contain a response card behind the front cover. Otherwise, you can mail your comments to the following address:

Cisco Systems, Inc., Document Resource Connection
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

9. Obtaining Technical Assistance

Cisco provides Cisco.com as a starting point for all technical assistance. Customers and partners can obtain documentation, troubleshooting tips, and sample configurations from online tools. For Cisco.com registered users, additional troubleshooting tools are available from the TAC website.

Cisco.com is the foundation of a suite of interactive, networked services that provides immediate, open access to Cisco information and resources at anytime, from anywhere in the world. This highly integrated Internet application is a powerful, easy-to-use tool for doing business with Cisco.

Cisco.com provides a broad range of features and services to help customers and partners streamline business processes and improve productivity. Through Cisco.com, you can find information about Cisco and our networking solutions, services, and programs. In addition, you can resolve technical issues with online technical support, download and test software packages, and order Cisco learning materials and merchandise. Valuable online skill assessment, training, and certification programs are also available.

Customers and partners can self-register on Cisco.com to obtain additional personalized information and services. Registered users can order products, check on the status of an order, access technical support, and view benefits specific to their relationships with Cisco.

To access Cisco.com, go to the following website:

http://www.cisco.com

Annex A: The Search/Sort Script

1. Install the following script on the flash of the router as ios-script.exp
2. Create an alias to use the script: **alias exec logger tclsh ios-script.exp**
3. Use the "logger" alias command to execute the script.

   A menu is presented for different searching options. Select one of the options by number and then select the order for sorting: chronological (c), reverse chronological (r), or both (b).

The Search/Sort Script:

```
#############################################################################
########################
#
# This script needs to be copied to the routers flash file system.
#
# Usage:
#
#  Router# tclsh ios-script.tcl
#
#
# When invoked, this script does the following:
#
#  1) Copies the logging buffer to a flash file "temp.fil".
#  2) Copies the syslog event entries without the header to a list.
#  2) Brings up a menu of sorting options.
#  3) Searches and sorts the list output as chosen on the menu and displays it.
#
#
#
# Dependencies:
#
#  1) Must have logging buffered enabled on the router and you must have accumulated
#  some log entries before running this script.
#  2) Must have "service timestamps log datetime year" configured on router.
#
#
#  Febuary 2011 -- Joe Baynard & Clyde Layton
#  Copyright (c) 2011 by cisco Systems, Inc.
#  All rights reserved.
#####################################################################
```

```
#############################################################################
#########
#NAME     createLogFile
#
#DESCRIPT
#        1. Creates file in flash from log called temp.fil
#
#
#AUTHOR   jbaynard@cisco.com
#Modified
#############################################################################
########

proc createLogFile { } {
   # execute show log and capture to variable $logBuffer
   set logBuffer [exec show log | redirect flash:temp.fil]
}

#############################################################################
#########
#NAME     removeBackspace
#
#DESCRIPT
#        1. Takes standard in and removes the back spaces
#
#
#AUTHOR   clayton@cisco.com
#Modified
#############################################################################
########

proc removeBackspace {input} {
   # Remove the back spaces from standard in
```

```
   set CNTL_H \010

   while { 1 } {
      set offset [string first $CNTL_H $input]
      if { $offset < 0 } {
         break
      }
      # Remove backspace character plus previous character
      set input "[string range $input 0 [expr {$offset - 2}]][string range $input [expr {$offset + 1}] end]"
   }

   # Return resulting string
   return $input
}

################################################################################
#########
#NAME      convertTime
#
#DESCRIPT
#         1. Takes user input for date or date time and converts it to a number
#
#
#AUTHOR    clayton@cisco.com
#Modified
################################################################################
########

proc convertTime {input} {
   # Takes the input and using the clock scan converts it to a number
   set output "error"
```

```
if {[catch [set output [clock scan "$input"]]] == 1} {

        return $output
    }

    set error "error"
    return $error
}


################################################################################
#########
#NAME      reverseResults
#
#DESCRIPT
#         1. Takes a variable and reverses the lines of data (bottom to top)
#
#
#AUTHOR    clayton@cisco.com
#Modified
################################################################################
########

proc reverseResults {results } {

    set task ""

    # Here we get the count of lines in the variable
    set results [split $results "\n"]
    set i [llength $results]

    # Here we start at the bottom and work our way up creating a new variable
    while {$i >= 0} {
```

```
        append task "[lindex $results $i]\n"
        incr i -1
    }
    return $task
}

#############################################################################
#########
#NAME      convertmonth
#
#DESCRIPT
#         1. Takes a variable containing the 3 letter abbreviation for a month
#           and converts it to a 2 digit number
#
#AUTHOR    clayton@cisco.com
#Modified
#############################################################################
########

proc convertmonth {month} {

    #This is used to verify the month is correct and change it into a number
    set months [list "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov"
"Dec"]
    set i 01
    foreach match $months {
        if {$month == $match} {
            return [format %02d "$i"]
        } else {
            incr i
        }
    }
    return "not found"
```

```
}

##############################################################################
#########
#NAME      getIPaddress
#
#DESCRIPT
#         1. Gets an IP address from user
#         2. Verifies the IP address
#         3. Confirms with user the ip address
#         4. Searches the temp.fil for matching IP address
#         5. Creates a variable with matching IP address
#         6. Displays results according to user preference
#
#AUTHOR    clayton@cisco.com
#Modified
##############################################################################
########

proc getIPaddress { } {
   set exit 0
   set do_check 0
   set confirm ""
   set get_ip 0
   set done 0

   # This while loop loops until the ip address is
   # entered correctly according to user
   while {$get_ip == 0} {

         puts "\n"
         puts -nonewline "Enter ip address : "
         flush stdout
```

```
        set ip_ent [gets stdin]
        set ip_ent [removeBackspace $ip_ent]

        if  [regexp  "(\[0-9\]{1,3}\\\.\[0-9\]{1,3}\\\.\[0-9\]{1,3}\\\.\[0-9\]{1,3})"  $ip_ent
ip_addr] {
            incr get_ip
            break
        } else {
            puts "\n"
            puts "Ip $ip_ent is not correct format"
            puts -nonewline "Do you want to exit y\/n : "
            flush stdout
            set confirm [gets stdin]
            set confirm [removeBackspace $confirm]
            if {[regexp "(\[yY\])" $confirm]} {
                return
            } else {
                set ip_addr ""
            }
        }
    }

    if {$exit != 0} {
        incr do_check
    }

    # This verifies that the user has indicated that the ip address
    # is correct and wants to do the search and does the searching
    if {$do_check == 0} {
        set complete 0
        set xx 0
        set num_ip 0
```

```
        catch {set infile [open "flash:temp.fil" r]}
        set logBuffer ""
        while {$xx >= 0} {
           set xx [gets $infile logline]

           if [regexp $ip_addr $logline] {

                 append logBuffer "$logline \n"
           }
        }
        catch {close $infile}

        # This gives the user a choice on how to display the data
        puts  -nonewline "Do you want your results in chronological order or reverse chronological order or both?  c\/r\/b : "
        flush stdout
        set confirm [gets stdin]
        set confirm [removeBackspace $confirm]
        if {[regexp "(\[cC\])" $confirm]} {
           puts "Here are the search results in chronological order:
**************************************************************************
**********************
$logBuffer
**************************************************************************
**********************
"
        } elseif {[regexp "(\[rR\])" $confirm]} {

           set logBuffer2 [reverseResults $logBuffer]
           puts "Here are the search results reverse chronological order:
**************************************************************************
**********************
$logBuffer2
```

```
**********************************************************************
**********************
"
        } else {
           set logBuffer2 [reverseResults $logBuffer]
           puts "Here are the search results in chronological order:
**********************************************************************
**********************
$logBuffer
**********************************************************************
**********************
"
           puts "Here are the search results reverse chronological order:
**********************************************************************
**********************
$logBuffer2
**********************************************************************
**********************
"
        }
   }
}

#########################################################################
#########
#NAME     getIPaddressRange
#
#DESCRIPT
#        1. Gets an begining and ending IP addresses from user
#        2. Verifies the IP addresses
#        3. Confirms with user the ip addresses
#        4. Searches the temp.fil for matching and address between IP address
#        5. Creates a variable with matching IP address results
#        6. Displays results according to user preference
```

```
#
#AUTHOR    clayton@cisco.com
#Modified
##############################################################################
########

proc getIpAddressRange { } {
   set ip_begin 0
   set ip_end 0
   set done 0
   set confirm ""
   set get_ips 0
   set get_begin 0
   set get_end 0
   set exit 0
   set do_check 0


   # This while loop loops until both the beginning
   # and ending ip addresses are entered correctly
   # and verified
   while {$done == 0} {

         # This while loop loops until both the beginning
         # and ending ip addresses are entered correctly
         # according to user
         while {$get_ips == 0} {

            # This while loop loops until the beginning
            # ip address is entered correctly
            while {$get_begin == 0} {
                  puts "\n"
                  puts -nonewline "Enter beginning ip address : "
```

```
            flush stdout
            set ip_begin1 [gets stdin]
            set ip_begin1 [removeBackspace $ip_begin1]

            if    [regexp    "(\[0-9\]{1,3}\\\.\[0-9\]{1,3}\\\.\[0-9\]{1,3}\\\.\[0-9\]{1,3})"
$ip_begin1 ip_begin] {
               incr get_begin
               break
            } else {
               puts "\n"
               puts "Beginning ip $ip_begin1 is not correct format"
               puts -nonewline "Do you want to exit y\/n : "
               flush stdout
               set confirm [gets stdin]
               set confirm [removeBackspace $confirm]
               if {[regexp "(\[yY\])" $confirm]} {
                     return
               } else {
                     set ip_begin ""
               }
            }
         }

         if {$exit != 0} {
            incr do_check
            break
         }

         # This while loop loops until the ending ip
         # address is entered correctly
         while {$get_end == 0} {
            puts "\n"
            puts -nonewline "Enter ending ip address :"
```

```
            flush stdout
            set ip_end1 [gets stdin]
            set ip_end1 [removeBackspace $ip_end1]

            if    [regexp    "(\[0-9\]{1,3}\\\.\[0-9\]{1,3}\\\.\[0-9\]{1,3}\\\.\[0-9\]{1,3})"
$ip_end1 ip_end] {
               incr get_end
               break
            } else {
               puts "\n"
               puts "Ending ip $ip_end1 is not correct format"
               puts -nonewline "Do you want to exit y\/n : "
               flush stdout
               set confirm [gets stdin]
               set confirm [removeBackspace $confirm]
               if {[regexp "(\[yY\])" $confirm]} {
                     return
               } else {
                     set ip_end ""
               }
            }
         }
         if {$exit != 0} {
            incr do_check
            break
         }

         # This verifies the beginning and ending ip addresses will work
         if {($ip_begin <= $ip_end)} {

            puts "\n"
            puts "You entered a starting ip of $ip_begin and a ending ip of $ip_end"
            puts  -nonewline "Is this correct y\/n : "
```

```
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]
                if {[regexp "(\[yY\])" $confirm]} {
                   incr get_ips
                   incr done
                   break
                } else {
                   set get_begin 0
                   set get_end 0
                }

            } else {
                puts "\n"
                puts "You entered a starting ip of $ip_begin which is after ending ip of
$ip_end"
                puts "\n"
                puts -nonewline "Do you wish to enter in new dates y\/n :"
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]

                if {[regexp "(\[yY\])" $confirm]} {
                   set get_begin 0
                   set get_end 0
                   break
                } else {
                   incr get_ips
                   incr done
                }
            }
        }
    }
```

```
   # This verifies that the user has indicated that the ip address
   # are correct and wants to do the search and does the searching
   if {$do_check == 0} {
         set complete 0
         set xx 0

         set num_ip_begin 0

         catch {set infile [open "flash:temp.fil" r]}
         set logBuffer ""
         while {$xx >= 0} {
            set xx [gets $infile logline]

            foreach check_value [split $logline "\ "] {

                  if    {[regexp    "(\[0-9\]{1,3}\\\.\[0-9\]{1,3}\\\.\[0-9\]{1,3}\\\.\[0-9\]{1,3})"
$check_value - check_ip temp]} {

                     set num_check_ip 0

                     if {($ip_begin <= $check_ip) && ($check_ip <= $ip_end)} {
                           append logBuffer "$logline \n"
                           break
                     }
                  }
            }
         }
         catch {close $infile}

         # This gives the user a choice on how to display the data
         puts  -nonewline "Do you want your results in chronological order or reverse
chronological order or both?  c\/r\/b : "
```

```
        flush stdout
        set confirm [gets stdin]
        set confirm [removeBackspace $confirm]
        if {[regexp "(\[cC\])" $confirm]} {
           puts "Here are the search results in chronological order:
**************************************************************************
**********************
$logBuffer
**************************************************************************
**********************
"
        } elseif {[regexp "(\[rR\])" $confirm]} {

           set logBuffer2 [reverseResults $logBuffer]
           puts "Here are the search results reverse chronological order:
**************************************************************************
**********************
$logBuffer2
**************************************************************************
**********************
"
        } else {
           set logBuffer2 [reverseResults $logBuffer]
           puts "Here are the search results in chronological order:
**************************************************************************
**********************
$logBuffer
**************************************************************************
**********************
"
           puts "Here are the search results reverse chronological order:
**************************************************************************
**********************
$logBuffer2
```

```
**********************************************************************
**********************
"
        }
   }
}

#############################################################################
#########
#NAME     getDateRange
#
#DESCRIPT
#        1. Gets an begining and ending date from user
#        2. Verifies the dates
#        3. Confirms with user the dates
#        4. Searches the temp.fil for matching and between the dates
#        5. Creates a variable with matching date results
#        6. Displays results according to user preference
#
#AUTHOR    clayton@cisco.com
#Modified
#############################################################################
########

proc getDateRange { } {
   set date_begin 0
   set date_end 0
   set done 0
   set confirm ""
   set get_dates 0
   set get_begin 0
   set get_end 0
   set exit 0
```

```
set do_check 0

# This while loop loops until both the beginning
# and ending dates are entered correctly
# and verified
while {$done == 0} {

    # This while loop loops until both the beginning
    # and ending dates are entered correctly
    # according to user
    while {$get_dates == 0} {

        # This while loop loops until the beginning
        # date is entered correctly
        while {$get_begin == 0} {
            puts "\n"
            puts -nonewline "Enter beginning date ie Feb 11, 2011 : "
            flush stdout
            set date_begin [gets stdin]
            set date_begin [removeBackspace $date_begin]

            if [regexp "(\[A-Z\]{1}\[a-z\]{2})(\ +)(\[0-9\]{1,2})(,\ +)(\[0-9\]{4})"
$date_begin - start_mon - start_day - start_year] {
                incr get_begin
                break
            } else {
                puts "\n"
                puts "Beginning date $date_begin is not correct format"
                puts -nonewline "Do you want to exit y\/n : "
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]
                if {[regexp "(\[yY\])" $confirm]} {
```

```
                return
            } else {
                set date_begin ""
            }
        }
    }

    if {$exit != 0} {
        incr do_check
        break
    }

    # This while loop loops until the ending
    # date is entered correctly
    while {$get_end == 0} {
        puts "\n"
        puts -nonewline "Enter ending date ie Feb 11, 2011: "
        flush stdout
        set date_end [gets stdin]
        set date_end [removeBackspace $date_end]

        if [regexp "(\[A-Z\]{1}\[a-z\]{2})(\ +)(\[0-9\]{1,2})(,\ +)(\[0-9\]{4})"
$date_end - end_mon - end_day - end_year] {
            incr get_end
            break
        } else {
            puts "\n"
            puts "Ending date $date_end is not correct format"
            puts -nonewline "Do you want to exit y\/n : "
            flush stdout
            set confirm [gets stdin]
            set confirm [removeBackspace $confirm]
            if {[regexp "(\[yY\])" $confirm]} {
```

```
                return
            } else {
                set date_end ""
            }
        }
    }
    if {$exit != 0} {
        incr do_check
        break
    }

    set month_found 0
    set month_bad 0

    # This verifies the beginning and ending month valid
    while {$month_found == 0} {
        set temp [convertmonth $start_mon]
        if {$temp == "not found"} {
            incr month_bad
            puts "\n"
            puts "You entered a invalid start month entry."
            puts "Valid months are Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
or Dec."
            puts "\n"
            puts -nonewline "Do you wish to enter in new dates y\/n :"
            flush stdout
            set confirm [gets stdin]
            set confirm [removeBackspace $confirm]

            if {[regexp "(\[yY\])" $confirm]} {
                set get_begin 0
                set get_end 0
                set get_dates 0
```

```
                break
            } else {
                incr get_dates
                incr done
                incr do_check
            }
        }
        set start_mon $temp
        set temp [convertmonth $end_mon]
        if {$temp == "not found"} {
            incr month_bad
            puts "\n"
            puts "You entered a invalid end month entry."
            puts "Valid months are Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
or Dec."
            puts "\n"
            puts -nonewline "Do you wish to enter in new dates y\/n :"
            flush stdout
            set confirm [gets stdin]
            set confirm [removeBackspace $confirm]

            if {[regexp "(\[yY\])" $confirm]} {
                set get_begin 0
                set get_end 0
                set get_dates 0
                break
            } else {
                incr get_dates
                incr done
                incr do_check
            }
        }
        set end_mon $temp
```

```
        incr month_found
    }

    # If beginning and ending month are good we convert the date time to seconds
    # and verify that they will search correctly
    if {$month_bad == 0} {
        set date_begin1 $date_begin
        set date_end1 $date_end
        set date_begin [convertTime $start_year-$start_mon-$start_day]
        set date_end [convertTime $end_year-$end_mon-$end_day]

        if {($date_begin == "error") || ($date_end == "error")} {
           puts "\n"
           puts "You entered a wrong date of $date_begin1 or $date_end1"
           puts  -nonewline "Do you wish to re-enter dates y\/n : "
           flush stdout
           set confirm [gets stdin]
           set confirm [removeBackspace $confirm]

           if {[regexp "(\[yY\])" $confirm]} {
                set get_begin 0
                set get_end 0
                break
           } else {
                return
           }
        }

        if {($date_begin <= $date_end)} {
           puts "\n"
           puts "You entered a starting date of $date_begin1 and a ending date of
$date_end1"
           puts  -nonewline "Is this correct y\/n : "
```

```
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]

                if {[regexp "(\[yY\])" $confirm]} {
                        incr get_dates
                        incr done
                        break
                } else {
                        set get_begin 0
                        set get_end 0
                }

            } else {
                puts "\n"
                puts "You entered a starting date of $date_begin1 which is after ending
date of $date_end1"
                puts "\n"
                puts -nonewline "Do you wish to enter in new dates y\/n :"
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]

                if {[regexp "(\[yY\])" $confirm]} {
                        set get_begin 0
                        set get_end 0
                        break
                } else {
                        incr get_dates
                        incr done
                }
            }
        }
```

```
        }
    }

    # If beginning and ending month are good and user
    # verified them do search
    if {$do_check == 0} {
        set complete 0
        set xx 0
        catch {set infile [open "flash:temp.fil" r]}
        set logBuffer ""
        while {$xx >= 0} {
            set xx [gets $infile logline]

            # This seaches for beginng date and enables the
            # between dates matching
            if {[regexp "(\[A-Z\]{1}\[a-z\]{2})(\ +)(\[0-9\]{1,2})(\ +)(\[0-9\]{4})" $logline
- start_check_mon - start_check_day - start_check_year]} {

                set temp_mon [convertmonth $start_check_mon]
                set start_check_mon $temp_mon
                set a [format %02d "$start_check_day"]
                set start_chech_day $a
                set temp1 [convertTime $start_check_year-$start_check_mon-
$start_check_day]
#               set temp1 [clock scan $start_check_year-$start_check_mon-
$start_check_day]

                if {($temp1 >= $date_begin) && ($temp1 <= $date_end)} {
                    append logBuffer "$logline \n"
                    set copy_from_begin 0
                }
            }
        }
```

```
        # This gives the user a choice on how to display the data
        puts  -nonewline "Do you want your results in chronological order or reverse chronological order or both?  c\/r\/b : "
        flush stdout
        set confirm [gets stdin]
        set confirm [removeBackspace $confirm]
        if {[regexp "(\[cC\])" $confirm]} {
           puts "Here are the search results in chronological order:
**************************************************************************
**********************
$logBuffer
**************************************************************************
**********************
"
        } elseif {[regexp "(\[rR\])" $confirm]} {

           set logBuffer2 [reverseResults $logBuffer]
           puts "Here are the search results reverse chronological order:
**************************************************************************
**********************
$logBuffer2
**************************************************************************
**********************
"
        } else {
           set logBuffer2 [reverseResults $logBuffer]
           puts "Here are the search results in chronological order:
**************************************************************************
**********************
$logBuffer
**************************************************************************
**********************
"
           puts "Here are the search results reverse chronological order:
```

```
**********************************************************************
**********************
$logBuffer2
**********************************************************************
**********************
"
        }
        catch {close $infile}
   }
}

#############################################################################
#########
#NAME     getDateTimeRange
#
#DESCRIPT
#        1. Gets an begining and ending date time from user
#        2. Verifies the dates and times
#        3. Confirms with user the dates and times
#        4. Searches the temp.fil for matching and between the dates and time
#        5. Creates a variable with matching date time results
#        6. Displays results according to user preference
#
#AUTHOR    clayton@cisco.com
#Modified
#############################################################################
########

proc getDateTimeRange { } {
   set date_begin 0
   set date_end 0
   set done 0
   set confirm ""
   set get_dates 0
```

```
set get_begin 0
set get_end 0
set exit 0
set do_check 0

# This while loop loops until both the beginning
# and ending date time are entered correctly
# and verified
while {$done == 0} {

    # This while loop loops until both the beginning
    # and ending date times are entered correctly
    # according to user
    while {$get_dates == 0} {

        # This while loop loops until the beginning
        # date time is entered correctly
        while {$get_begin == 0} {
            puts "\n"
            puts -nonewline "Enter beginning date ie Feb 11, 2011 00:00:00 : "
            flush stdout
            set date_begin [gets stdin]
            set date_begin [removeBackspace $date_begin]
            if [regexp "(\[A-Z\]{1}\[a-z\]{2})(\ +)(\[0-3\]{1}\[0-9\]{1})(,\ +)(\[0-9\]{4})(\ +)(\[0-2\]{1}\[0-9\]{1}:\[0-5\]{1}\[0-9\]{1}:\[0-5\]{1}\[0-9\]{1})" $date_begin - start_mon - start_day - start_year - start_time] {
                incr get_begin
                break
            } else {
                puts "\n"
                puts "Beginning date $date_begin is not correct format"
                puts -nonewline "Do you want to exit y\/n : "
                flush stdout
```

```
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]
                if {[regexp "(\[yY\])" $confirm]} {
                        return
                } else {
                        set date_begin ""
                }
        }
}

if {$exit != 0} {
        incr do_check
        break
}

# This while loop loops until the ending
# date time is entered correctly
while {$get_end == 0} {
        puts "\n"
        puts -nonewline "Enter ending date ie Feb 11, 2011 00:00:00 :"
        flush stdout
        set date_end [gets stdin]
        set date_end [removeBackspace $date_end]

        if [regexp "(\[A-Z\]{1}\[a-z\]{2})(\ +)(\[0-9\]{1,2})(,\ +)(\[0-9\]{4})(\
+)(\[0-2\]{1}\[0-9\]{1}:\[0-5\]{1}\[0-9\]{1}:\[0-5\]{1}\[0-9\]{1})" $date_end - end_mon
- end_day - end_year - end_time] {
                incr get_end
                break
        } else {
                puts "\n"
                puts "Ending date $date_end is not correct format"
                puts -nonewline "Do you want to exit y\/n : "
```

```
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]
                if {[regexp "(\[yY\])" $confirm]} {
                        return
                } else {
                        set date_end ""
                }
        }
}
if {$exit != 0} {
        incr do_check
        break
}

set month_found 0
set month_bad 0

# This verifies the beginning and ending month valid
while {$month_found == 0} {
        set temp [convertmonth $start_mon]
        if {$temp == "not found"} {
                incr month_bad
                puts "\n"
                puts "You entered a invalid start month entry."
                puts "Valid months are Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
or Dec."
                puts "\n"
                puts -nonewline "Do you wish to enter in new dates y\/n :"
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]
                if {[regexp "(\[yY\])" $confirm]} {
```

```
                set get_begin 0
                set get_end 0
                set get_dates 0
                break
            } else {
                return
            }
        }
        set start_mon $temp
        set temp [convertmonth $end_mon]
        if {$temp == "not found"} {
            incr month_bad
            puts "\n"
            puts "You entered a invalid end month entry."
            puts "Valid months are Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
or Dec."
            puts "\n"
            puts -nonewline "Do you wish to enter in new dates y\/n :"
            flush stdout
            set confirm [gets stdin]
            set confirm [removeBackspace $confirm]

            if {[regexp "(\[yY\])" $confirm]} {
                set get_begin 0
                set get_end 0
                set get_dates 0
                break
            } else {
                return
            }
        }
        set end_mon $temp
        incr month_found
```

```
        }

        # If beginning and ending month are good we convert the date time to seconds
        # and verify that they will search correctly
        if {$month_bad == 0} {
            set date_begin1 $date_begin
            set date_end1 $date_end
#           set date_begin [clock scan "$start_year-$start_mon-$start_day $start_time"]
#           set date_end [clock scan "$end_year-$end_mon-$end_day $end_time"]
            set date_begin [convertTime "$start_year-$start_mon-$start_day $start_time"]
            set date_end [convertTime "$end_year-$end_mon-$end_day $end_time"]

            if {($date_begin == "error") || ($date_end == "error")} {
               puts "\n"
               puts "You entered a wrong date of $date_begin1 or $date_end1"
               puts  -nonewline "Do you wish to re-enter dates y\/n : "
               flush stdout
               set confirm [gets stdin]
               set confirm [removeBackspace $confirm]

               if {[regexp "(\[yY\])" $confirm]} {
                    set get_begin 0
                    set get_end 0
                    break
               } else {
                    return
               }
            }

            if {($date_begin <= $date_end)} {
               puts "\n"
```

```
                puts "You entered a starting date of $date_begin1 and a ending date of
$date_end1"
                puts  -nonewline "Is this correct y\/n : "
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]
                if {[regexp "(\[yY\])" $confirm]} {
                        incr get_dates
                        incr done
                        break
                } else {
                        set get_begin 0
                        set get_end 0
                }

              } else {
                puts "\n"
                puts "You entered a starting date of $date_begin1 which is after ending
date of $date_end1"
                puts "\n"
                puts -nonewline "Do you wish to enter in new dates y\/n :"
                flush stdout
                set confirm [gets stdin]
                set confirm [removeBackspace $confirm]

                if {[regexp "(\[yY\])" $confirm]} {
                        set get_begin 0
                        set get_end 0
                        break
                } else {
                        incr get_dates
                        incr done
                }
```

```
                }
            }
        }
    }

    # If beginning and ending month are good and user
    # verified them do search
    if {$do_check == 0} {
        set complete 0
        set xx 0
        catch {set infile [open "flash:temp.fil" r]}
        set logBuffer ""
        while {$xx >= 0} {
            set xx [gets $infile logline]

            # This seaches for beginng date time and enables the
            # between date time matching
            if {[regexp "(\[A-Z\]{1}\[a-z\]{2})(\ +)(\[0-9\]{1,2})(\ +)(\[0-9\]{4})(\ +)(\[0-6\]{1}\[0-9\]{1}:\[0-6\]{1}\[0-9\]{1}:\[0-6\]{1}\[0-9\]{1})" $logline - start_check_mon - start_check_day - start_check_year - start_check_time]} {

                set temp_mon [convertmonth $start_check_mon]
                set start_check_mon $temp_mon
                set a [format %02d "$start_check_day"]
                set start_chech_day $a
                set temp1 [clock scan "$start_check_year-$start_check_mon-$start_check_day $start_check_time"]

                if {($temp1 >= $date_begin) && ($temp1 <= $date_end)} {
                    append logBuffer "$logline \n"
                    set copy_from_begin 0
                }
            }
```

```
        }

        # This gives the user a choice on how to display the data
        puts  -nonewline "Do you want your results in chronological order or reverse chronological order or both?  c\/r\/b : "
        flush stdout
        set confirm [gets stdin]
        set confirm [removeBackspace $confirm]
        if {[regexp "(\[cC\])" $confirm]} {
           puts "Here are the search results in chronological order:
**************************************************************************
**********************
$logBuffer
**************************************************************************
**********************
"
        } elseif {[regexp "(\[rR\])" $confirm]} {

           set logBuffer2 [reverseResults $logBuffer]
           puts "Here are the search results reverse chronological order:
**************************************************************************
**********************
$logBuffer2
**************************************************************************
**********************
"
        } else {
           set logBuffer2 [reverseResults $logBuffer]
           puts "Here are the search results in chronological order:
**************************************************************************
**********************
$logBuffer
**************************************************************************
**********************
"
```

```
            puts "Here are the search results reverse chronological order:
**************************************************************************
**********************
$logBuffer2
**************************************************************************
**********************
"
        }
    }
    catch {close $infile}
}


#############################################################################
#########
#NAME      getBufferedLog
#
#DESCRIPT
#         1. Takes the temp.fil
#         2. Removes the header of the log information
#         3. Creates and returns logList with data converted to 2 variable list
#
#AUTHOR    jbaynard@cisco.com
#Modified
#############################################################################
########


proc getBufferedLog { } {
    set index 0
    set logline ""
    set logList ""

    catch {set infile [open "flash:temp.fil" r]}
```

```
    set logBuffer ""

    #
    # Create a list from the log buffer dividing the timestamp and log entries into
    # two separate logical fields separating them at the first '%' character.
    #
    set xx 0
    set headerComplete 0
    set header ""

    while {$xx >= 0} {
      set xx [gets $infile logline]

      if {$headerComplete == 0} {
        append header "$logline \n"
      }
      if [regexp "Log Buffer" $logline] {
        set headerComplete 1
      }
      set logline [string trim $logline]
      if {$logline > "" && [regexp "%" $logline] == 1} {
       incr index
        lappend logList   "\{[string range $logline 0 [expr [string first "%" $logline ]-1]]\}
\{[string range $logline [expr [string first "%" $logline ]] 255] \}"
      }
    }
    catch {close $infile}

    return $logList
}
```

```
#
# SCRIPT ENTRY POINT
#

# Create the temp log file
puts "Copying log file to flash.This may take a couple of minutes."
createLogFile

# Read the log file and filter the events
set fixedLog [getBufferedLog]
set finished 0
set get_task 0

while {$finished == 0} {

   while {$get_task == 0} {

         puts "\n\n\n"
         puts
"=============================================================================
====="
         puts " Syslog event search: "
         puts
"=============================================================================
====="
         puts " List only log events containing an IP Address.....1 "
         puts " List only log events containing a range of IP Address.....2 "
         puts " List only log events within a range of dates.....3 "
         puts " List only log events within a date and time range.....4 "
         puts " List all log events in Reverse Chronological Order.....5 "
         puts " Exit.....6 "
```

```
        puts
"=============================================================================
====="
        puts "\n\n You will have to perform the following from the configuration mode"
        puts "for these to work"
        puts "service timestamps log datetime year"

        puts -nonewline "Enter choice: 1 - 6 : "
        flush stdout
        set choice [gets stdin]
        set choice [removeBackspace $choice]

        if {($choice > 6) || ($choice < 1)} {
           puts -nonewline "Not a valid choice"
           flush stdout
        } else {
           incr get_task
        }
   }

   # menu selection 1, match on IP address
   if {$choice == 6} {

        puts  -nonewline "Are you sure you want to exit y\/n : "
        flush stdout
        set confirm [gets stdin]
        set confirm [removeBackspace $confirm]
        if {[regexp "(\[yY\])" $confirm]} {
           incr finished
           incr get_task
           break
        } else {
           set choice 0
```

```
        set get_task 0
    }
}

if {$choice == 1} {
    getIPaddress
    set choice 0
}
if {$choice == 2} {
    getIpAddressRange
    set choice 0
}
if {$choice == 3} {
    getDateRange
    set choice 0
}
if {$choice == 4} {
    getDateTimeRange
    set choice 0
}
if {$choice == 5} {

    set complete 0
    set xx 0
    set reverse_log ""
    catch {set infile [open "flash:temp.fil" r]}
    set logBuffer ""
    while {$xx >= 0} {
        set xx [gets $infile logline]

        if {[regexp "(\[A-Z\]{1}\[a-z\]{2}\ +\[0-9\]{1,2}\ +\[0-9\]{4})" $logline]} {
            append reverse_log "$logline \n"
        }
```

```
        }
        catch {close $infile}
        set reverse_log [reverseResults $reverse_log]
        puts "Here are the search results reverse chronological order:
**************************************************************************
**********************
$reverse_log
**************************************************************************
**********************
"


   }

   puts "\n"
   puts  -nonewline "Would you like to do another search? y\/n : "

   flush stdout
   set confirm [gets stdin]
   set confirm [removeBackspace $confirm]
   if {[regexp "(\[nN\])" $confirm]} {
        incr finished
   } else {
        set get_task 0
        set choice 0
   }
}

# delete the temp file
file delete flash:temp.fil
puts "\nDone!"
```