Cisco IOS Software Modularity on Cisco Catalyst 6500 Series Switches

This document provides a detailed technical insight into the benefits, features, and capabilities of Cisco IOS® Software Modularity on Cisco® Catalyst® 6500 Series switches.

INTRODUCTION

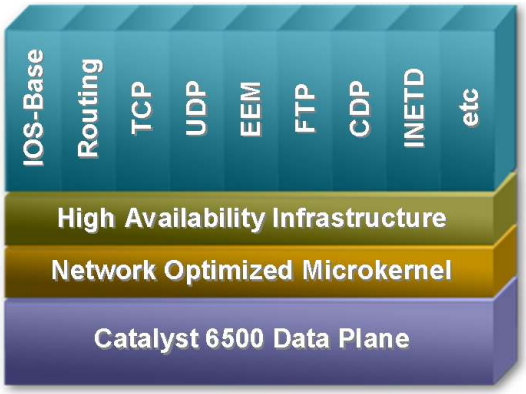
Today's networking devices have to offer maximum uptime to provide service to mission-critical traffic such as voice, video, and data applications. Although high availability is usually built into the core and distribution layers through redundant systems, making a single networking device more resilient becomes most important in the wiring closet, data center access, and WAN edge of an enterprise network because these are typically single points of failure for connected end devices. Metro Ethernet Access networks have a similar requirement in order to meet strict service-level agreements (SLAs) made to customers. Because business depends on network applications, availability and reliability are indispensable for network operators and administrators.

The Cisco Catalyst 6500 Series with Cisco IOS Software Modularity minimizes downtime and boosts operational efficiency through evolutionary software infrastructure advancements. By enabling modular Cisco IOS Software subsystems to run as independent processes, this innovation minimizes unplanned downtime through self-healing processes; simplifies software changes through subsystem In-Service Software Upgrades (ISSU); and helps enable process-level, automated policy control by integrating Embedded Event Manager (EEM).

CISCO IOS SOFTWARE MODULARITY FOR THE CISCO CATALYST 6500 SERIES

Cisco IOS Software is designed to meet the demanding IP services and control-plane scalability requirements of evolving networks. Cisco IOS Software consists of hundreds of subsystems, each of which defines part of a technology, that run in a shared memory space in order to maximize software forwarding performance.(See Figure 1.)

Figure 1. Architecture of Cisco IOS Software Modularity Showing Control-Plane and Data-Plane Separation and Independent Processes on the Cisco Catalyst 6500 Series



The Cisco Catalyst 6500 Series delivers hardware-based forwarding through application-specific integrated circuits (ASICs) on a central policy feature card (PFC) or distributed forwarding cards (DFCs). The control-plane functions on the Catalyst 6500 Series run on dedicated CPUs on the Multilayer Switch Feature Card (MSFC) complex.

- **Control plane**—Manages control traffic such as routing protocols, management traffic, and so on
- **Data plane**—Is responsible for the actual forwarding of traffic using ASICs

A completely separate data plane helps ensure that traffic forwarding continues even if there is a disruption in the control plane, as long as the software is intelligent enough to program the hardware for nonstop operation. With Cisco Catalyst 6500 Series Supervisor Engine redundancy, the Non-Stop Forwarding (NSF) and Stateful Switchover (SSO) features available on the Catalyst 6500 Series provide a continuous data plane even in the event of a hardware failure on the active supervisor engine. The need for fault isolation and decoupling of the control and data plane results in a shift of focus at the OS level. Specifically, changes or issues in the control-plane software should not affect forwarding on the data plane.

Cisco IOS Software Modularity combines subsystems into individual processes and enhances the memory architecture to provide process-level fault isolation and subsystem ISSU capability. These enhancements are delivered on Cisco IOS Software for the Cisco Catalyst 6500 Series Supervisor Engine 720 and Cisco Catalyst 6500 Supervisor Engine 32, maintaining the feature richness and operational environment with which network operators are familiar. Cisco first introduced IOS Software Modularity in maintenance release 12.2(18)SXF4.

Operational Consistency

Although Cisco IOS Software Modularity adds many enhancements to Cisco IOS Software on the Catalyst 6500 Series switches, no changes from an operational point of view are necessary. The command-line interface (CLI) as well as management-related interfaces such as Simple Network Management Protocol (SNMP) or Syslog are the same as before. New commands to exec and configuration mode as well as show commands have been added to support the new capability. Software releases and rebuilds remain the same, with additional support for subsystem patching.

Protected Memory

Cisco IOS Software Modularity helps enable a memory architecture where processes make use of a protected address space. Each process and its associated subsystems “live” in an individual memory space. Processes can only communicate using defined Inter Process Communication (IPC). Using this paradigm, memory corruption across process boundaries becomes virtually impossible. Communications between subsystems within the same process can happen directly, delivering consistent control-plane performance.

Fault Containment

The benefit of protected memory spaces is increased availability because problems occurring in one process cannot affect other parts of the system. For example, if a less critical system process fails or is found not to operate as expected, critical functions required to maintain packet forwarding are not affected. More specifically if UDP.proc was to fail only features dependent on UDP processing are affected.

Process Restartability

Building on the protected memory space and fault containment, the modular processes are now individually restartable. For test purposes or nonresponding processes, a CLI is provided to manually restart processes. This allows fast recovery from transient errors without the need to disrupt forwarding.

While manual restarting of processes is important, it is crucial to check state and health of processes continuously. An integrated high-availability subsystem takes care of this task by constantly checking the state of processes and keeping track

of how many times a process restarted in a defined time interval. This high-availability subsystem provides the capability of restarting processes in order to recover as fast as possible from various faults. In the event a process restart does not restore the system, the high-availability subsystem will take more drastic actions such as initiating a supervisor-engine switchover or a system restart.

Because each process has its own protected environment, check-pointing of state information is available as needed. The check-pointing architecture allows maintaining this information during a process restart or a failover. The high-availability subsystem can make use of this information in the event of a process restart to provide a stateful recovery. When a process restarts, check-pointed information (for example, state of the Intermediate System-to-Intermediate System [IS-IS] routing protocol information) will be used by the restarted process to recover as fast as possible. The check-pointed information is only made available during the first restart of a process in a specified time interval to help ensure that the checkpointed state information itself is not causing errors.

Modularized Processes

Several control-plane functions have been modularized to cover the most commonly used features. Examples of modular processes include but are not limited to IP Routing (RIP, EIGRP, OSPF, IS-IS, BGP), TCP, UDP, CDP, Embedded Event Manager, Installer, etc.

Because the architecture allows further modularization of Cisco IOS Software, the evolution of Software Modularity will increase the number of independent modular processes.

Subsystem In-Service Software Upgrades

Undoubtedly, the most important benefit of the protected memory space as well as the restartability of processes is the ability to make changes to a system during runtime. Cisco IOS Software Modularity enhances the Cisco IOS Software infrastructure to allow selective system maintenance through individual patches. A patch is a single fix that can affect one or multiple subsystems. Patches will be initially delivered in the form of Maintenance Packs-similar to Software Maintenance Updates (SMUs) in Cisco IOS XR Software for the Carrier Routing System (CRS-1) and the Cisco 12000 XR Series routers. A Maintenance Pack can include one or more patches that can be applied during runtime. The initial release of Software Modularity will focus on the delivery of security advisories reported by the Product Security Incident Reporting Team (PSIRT).

By providing versioning and patch-management capabilities, Maintenance Packs can be downloaded, verified, installed, and activated without the need to restart the system. Depending on what part of the OS needs to be patched, the network operator now has the flexibility to introduce software changes at any time. Most patches changing modular processes will not require a supervisor failover or system restart.. A patch only affects the components required to fix a particular software issue, and therefore the code certification time is significantly reduced. A network administrator now only has to verify the portion of the software associated with the fix. After the decision to apply a Maintenance Pack has been made, multiple Cisco Catalyst 6500 Series switches throughout the network can be upgraded quickly using the subsystem ISSU feature.

INTERACTION OF SOFTWARE MODULARITY AND THE CISCO CATALYST 6500 SERIES HARDWARE ARCHITECTURE

The Cisco Catalyst 6500 Series switches make use of ASICs to perform forwarding of traffic in hardware. Control and management tasks such as routing protocols, terminal sessions (Telnet, Secure Shell [SSH] Protocol, etc.), SNMP, and others need to be performed by the CPU of the switch. Therefore the terms “data plane” and “control plane” are used to differentiate between two parts of the system. Because the data plane needs to get the correct entries for performing the hardware forwarding it interfaces with the control plane, which learns this information by using routing protocols. After these entries are programmed in hardware, only updates need to be sent from the control plane to the data plane.

Combining Software Modularity with this type of architecture and the NSF capability of routing protocols like Enhanced Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First (OSPF), IS-IS, and Border Gateway Protocol (BGP) has a significant benefit. Even a system with a single supervisor engine can make use of the NSF capability[1]. The restart capability of Software Modularity allows restarting any modular process such as the IP routing process while the rest of the system continues to operate. The following sequence explains how a system would recover without any user interaction:

- As a starting point, all routing protocols (control plane) have converged and traffic is being forwarded by the data plane
- The IP routing process fails
- While the process is being restarted the data plane keeps on forwarding traffic because it keeps its current hardware entries
- IP routing process recovers and sends an NSF message to its neighbors telling that it is recovering
- NSF-aware neighbors will keep the entries they learned from the recovering system in their table and send their information back to the recovering system
- When the control plane has processed all routing updates it received from its neighbors, it will program the changes to the data plane.

Note: Forwarding on the data plane is being performed at all times while this process is happening. It is, however, important to understand that this implementation does not protect against hardware failures of the supervisor engine. To recover from a hardware failure a secondary supervisor engine is still required.

MODES OF OPERATION

Cisco IOS Software images offering Software Modularity can run in two different modes-either using a single binary image or an installed image. Running the system from a binary image is what most operators are familiar with: a single binary Cisco IOS Software image serving the switch processor and route processor. In contrast, the system will run from a file system with a directory and file structure in if it is running an installed image. The suggested mode of operation is “installed mode” since it allows making use of subsystem ISSU and offers more efficient use of memory.

Differences

Table 1 outlines the differences between the two modes of operation.

Table 1. Differences in Modes of Operation

	Installed Image	Single Binary Image
Protected Memory Space	X	X
Fault Containment	X	X
Process Restartability	X	X
Process Check-Pointing	X	X
Subsystem ISSU (Patching)	X	-
Repackaging of Image with Patches	X	N/A
More Efficient Use of Memory	X[2]	-

1 For details on the NSF implementation of the Cisco Catalyst 6500 Series, please consult the product literature/white paper section at http://www.cisco.com/go/catalyst6500.
2 At startup only necessary line card code is loaded in DRAM. When a new line card type is inserted, the appropriate code is loaded dynamically.

The repackaging capability allows the network administrator to compress the base image and all applied patches as well as the whole patching history including tags into one binary file. This repacking feature will be explained in more detail later.

A system running from an installed image allocates memory for line cards based on the insertion and removal of these. There is a requirement that the file system be present at all times. In case external compact flash memory is used, the compact flash must not be removed during runtime. While a system running from a binary image loads all code during boot-up time, systems running from an installed image do so as needed for configured features during runtime.

Installing an Image

Installing an image is equivalent to creating a directory structure and decompressing the binary file into that layout. This action is performed by the installer, which is a component added to images offering Cisco Software Modularity.

When running an image with Cisco IOS Software Modularity, the source location of the new image can be either local or remote. Options to install an image from remote include Trivial File Transfer Protocol (TFTP), FTP, and Remote Copy Protocol (RCP).

To install the image use the following command:

```
6500#install file ?
bootdisk: Source URL for software to install
bootflash: Source URL for software to install
disk0: Source URL for software to install
disk1: Source URL for software to install
ftp: Source URL for software to install
rcp: Source URL for software to install
scp: Source URL for software to install
sup-bootdisk: Source URL for software to install
sup-bootflash: Source URL for software to install
tftp: Source URL for software to install

6500#install file ftp: ?
disk0: Install software to search root with prefix disk0:
disk1: Install software to search root with prefix disk1:
sup-bootdisk: Install software to search root with prefix sup-bootdisk:
sup-bootflash: Install software to search root with prefix sup-bootflash:

6500#install file ftp: disk0:/sys
Address or name of remote host []? 172.16.1.1
Source filename []? s72033-ipservicesk9-vz.bin
!!!!!!!!!!!!!!!!
```

The system will then start copying the compressed image to the memory, decompress it, validate checksum, and then write it to the file system specified. The keyword interactive can be specified to get more detailed output.

Provided there is enough space available, a file system can also host multiple versions of Cisco IOS Software Modularity. Previously this was done by copying multiple files to a media. Because an installed image is comprised of a directory structure containing multiple files, the differentiator for an installed image is the topmost directory (in the previous example this would be the /sys directory). The system offers three locations per file system where images can be installed. These are:

/sys, /newsys, /oldsys. A new release can be installed to the /newsys location and tested. When certified, the previous software base could be moved to the /oldsys location and the new base could be moved from /newsys to /sys.

Changes to the Boot Variable and Binding an Image

As other Cisco Systems® devices are running Cisco IOS Software, the Catalyst 6500 makes use of a ROMMON which initializes the hardware and searches for a file (i.e. the Cisco IOS Software image) to boot. The file that shall be booted from ROMMON is defined with the "system boot flash …" command. As soon as the user saves the configuration this information is also written to the BOOT variable, which is accessible from ROMMON.

So far images were defined by pointing to the Cisco IOS Software binary file to specify the boot path. As described before Cisco IOS Software Modularity images can be run in two modes. While the behavior in single binary mode doesn't change (i.e. the administrator configures the bootstring to point to the binary), installed mode introduces some changes.

When a system is running from an installed image, a directory structure is created. The directory tree and its files are administered by the OS itself and the user does not need to have any knowledge of it. A side effect of this is that the bootvar variable needs to be defined using a different approach.

To simplify the process of defining the bootvar variable and at the same time make it secure and reliable, the concept of "binding" is introduced. When installing an image, a destination like disk0:/sys is specified. Once the image is installed, the system has to be bound to the location (let ROMMON know where to boot the system from). Binding a system to a location in return automatically defines the appropriate boot string. Here is an example:

This configuration command defines where the OS is located.

```
6500(config)#install bind disk0:/sys
```

To verify what has been added to the configuration, issue the following command.

Note: The whole boot string has been inserted. Although the boot string is longer compared to a system running from a single binary image, the user does not need to type in the boot string but just specifies the directory.

```
6500# show running-config | include disk0:
boot system disk0:/sys/s72033/base/s72033-ipservicesk9-vm
```

The network administrator can verify at any time from what location the system is running. This can be done using the show install running command.

As mentioned previously, multiple "installed" releases can reside on a single media. The differentiating factor in this case is the directory where they have been installed or extracted to. To change the location where the system boots from, the no install bind <location> configuration command would be used to remove the old entry. The new location could then be specified as shown in the previous example.

File System Requirements

Besides the need for more flash storage space, Cisco IOS Software Modularity does not have any special requirements with regards to file systems. To satisfy the need for additional flash space, compact flash media can be used. Cisco Systems is also offering an internal compact flash adapter that can be installed as a replacement of the internal switch processor linear flash (also known as sup-bootflash). Having an internal flash eliminates the risk of removal that an external compact flash is susceptible to. The file system performance has also been enhanced-for example, the time-consuming squeezing operation is no longer required. Cisco IOS Software Modularity supports ATA-based file systems only (for example, Compact Flash).

To benefit from the internal compact flash adapter (part number WS-CF-UPG=), the system must run a minimum ROMMON version of 8.4(2) for the switch processor. The notation for the file system will then change from sup-bootflash: to sup-bootdisk:. The Supervisor Engine 720 allows upgrading the ROMMON code without the need for physical access to the system. It can be done remotely and does not require any changes to the hardware. For details, please consult the ROMMON release notes on Cisco.com.

When the system is running from an installed image, the file system must be present at all times. Because line card code is loaded dynamically from the file system, lack of the file system could lead to problems.

For systems running from an installed image, the minimum storage capacity is 256 MB. 512 MB is the recommended media size.

PROCESS MANAGEMENT

Cisco IOS Software Modularity allows restarting of modular processes. As a general rule, users do not need to restart processes. The integrated high-availability subsystem continuously monitors the state of all processes and automatically initiates a restart of a process if needed.

The status of a process can be checked at any time. The following output shows the status of the IP routing process iprouting.iosproc

```
6500#show processes detailed iprouting.iosproc
Job Id: 68
PID: 16427
Executable name: iprouting.iosproc
Executable Path: sbin/iprouting.iosproc
Instance ID: 1
Respawn: ON............//Once this process crashes it will restart automatically//
Respawn count: 1........//This process has been started once (at bootup time)//
Respawn since last patch: 1
Max. spawns per minute: 30
Last started: Mon Aug 29 08:15:00 2005
Process state: Run
Feature name: iprouting
Core: SHAREDMEM MAINMEM
Max. core: 0
Level: 100
Mandatory: ON..........//This process has to be active; the system requires it to run//
Last restart userid:
Related Processes:
PID TID Stack pri state Blked HR:MM:SS:MSEC FLAGS NAME
16427 1 28K 10 Receive 1 0:00:00:0004 00000000 iprouting.iosproc
16427 2 28K 10 Receive 1 0:00:00:0000 00000000 iprouting.iosproc
16427 3 28K 10 Receive 1 0:00:00:0116 00000000 iprouting.iosproc
16427 4 28K 11 Nanosleep 0:00:00:0000 00000000 iprouting.iosproc
16427 5 28K 10 Receive 1 0:00:00:0000 00000000 iprouting.iosproc
----------------------------------------------------------------
6500#
```

Restarting of a process can be initiated by the user and should be used with caution. The following output shows the use of the “process restart <process name>“ command:

```
6500#process restart iprouting.iosproc
Restarting process iprouting.iosproc
6500#
```

When the process has restarted (this happens instantaneously) looking at the process details shows the following output.

```
6500#show processes detailed iprouting.iosproc
Job Id: 68
PID: 20523.........//New process ID has been assigned //
Executable name: iprouting.iosproc
Executable Path: sbin/iprouting.iosproc
Instance ID: 1
Respawn: ON
Respawn count: 2............//Counter has been increase to reflect process restart//
Respawn since last patch: 2.....................//Shows the number of process restarts
since the last patch was applied to that process//
Max. spawns per minute: 30
Last started: Mon Aug 29 08:16:00 2005
Process state: Run (last exit due to SIGTERM).......//Indicates why this process got
restarted//
Feature name: iprouting
Core: SHAREDMEM MAINMEM
Max. core: 0
Level: 100
Mandatory: ON
Last restart userid: cisco...//Indicates what user did initiate the last process restart//
Related Processes:
PID TID Stack pri state Blked HR:MM:SS:MSEC FLAGS NAME
20523 1 32K 10 Receive 1 0:00:00:0000 00000000 iprouting.iosproc
20523 2 32K 10 Receive 1 0:00:00:0000 00000000 iprouting.iosproc
20523 3 32K 10 Receive 1 0:00:00:0096 00000000 iprouting.iosproc
20523 4 32K 11 Nanosleep 0:00:00:0000 00000000 iprouting.iosproc
20523 5 32K 10 Receive 1 0:00:00:0016 00000000 iprouting.iosproc
---------------------------------------------------------------
6500#
```

PATCHING (Subsystem ISSU)

While restarting processes is an important mechanism allowing fault recovery without a system restart, its primary purpose is to allow the introduction of changes to a system during runtime. These subsystem level software changes are referred to as Subsystem ISSU or patching. For example, patching can be used to update the system with a single fix or minor changes in functionality. With the initial release of Cisco IOS Software Modularity, patches will be provided in the form of Maintenance Packs for publicly announced security vulnerabilities (PSIRT alerts) only. Future releases may introduce patching support for any type of fix and possibly even new functionality.

After a patch is installed, all processes to which the subsystem code belongs to need to be restarted to get the changes applied to the system. If check-pointing information is available it will be used to guarantee minimum disruption. In the case of

routing protocols, the NSF mechanism will prevent neighbors from removing the routes learned through the recovering system.

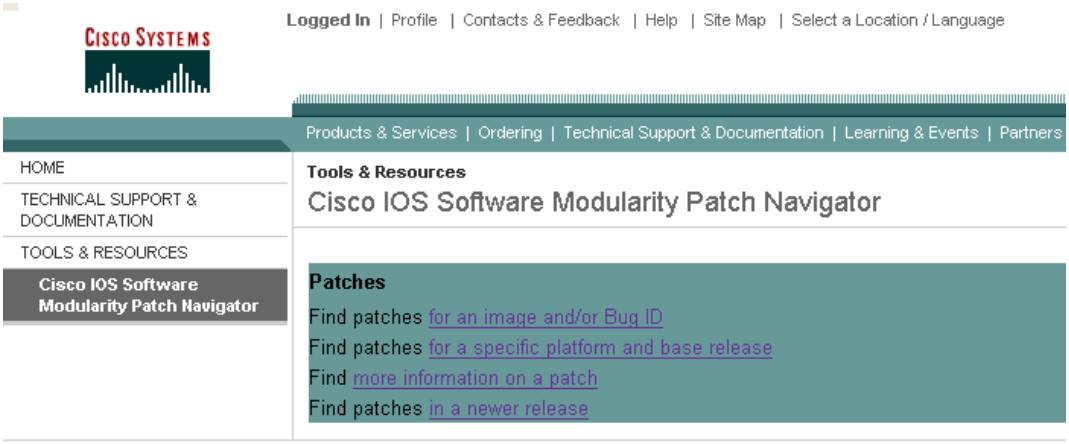
The following shows the process of applying a patch to a system. Please note that this is only applicable to systems running from an installed image.

1. First check what kind of base image and patches are running on the system. This can be verified by the show install running command:

```
6500#show install running
Software running on card installed at location s72033_rp - Slot 5 :

B/P C State Filename
--- - ------ --------
B Active disk0:/sys/s72033_rp/base/DRACO2_MP

Software running on card installed at location s72033 - Slot 5 :

B/P C State Filename
--- - ------ --------
B Active disk0:/sys/s72033/base/s72033-adventerprisek9_wan-vm

LEGEND:
-------:
B/P' - (B)ase image or (P)atch
`C' - (C)omitted
Pruned - This file has been pruned from the system
Active - This file is active in the system
PendInst - This file is set to be made available to run on the
system after next activation.
PendRoll - This file is set to be rolled back after next activation.
InstPRel - This file will run on the system after next reload
RollPRel - This file will be removed from the system after next reload
RPRPndIn - This file is both rolled back pending a reload, and pending
installation. On reload, this file will not run and will move to
PendInst state. If `install activate' is done before reload, pending
removal and install cancel each other and file simply remains active
IPRPndRo - This file is both installed pending a reload, and pending rollback.
If the card reloads, it will be active on the system pending a rollback
If `install activate' is done before a reload, the pending install and
removal with cancel each other and the file will simply be removed
6500#
```

2. The Maintenance Pack needs to be downloaded from Cisco.com[3]. Finding the right Maintenance Pack or patch for a specific image is very easy. Point your browser to http://www.cisco.com/go/pn (Figure 2). The following screen capture shows the patch navigator and the option how a patch can be found.

Figure 2. Software Modularity Patch Navigator



3. The Maintenance Pack/patch needs to be copied to the system and introduced into the file system. This is done using the installer, which copies the file to the system, verifies if the patch is applicable, and inserts it to the file system structure. Because the administration of the file system is done by the installer, there is no training required to handle any file system specific operations. The Maintenance Pack/patch can either manually be copied to any file system on the system or accessed using TFTP, FTP, or RCP. Already during the first step of the patching process, the installer will point out what process will need to be restarted. Here is some sample output:

```
6500#install file tftp://172.16.1.1/s72033-Yakhurana-00.pikespeak.pk_ptch disk0:/sys
Address or name of remote host [172.16.1.1]?
Source filename [s72033-Yakhurana-00.pikespeak.pk_ptch]?
!!!!!!!!!!!!
Verifying checksums of extracted files

Verifying installation compatibility
Gathering information for slot s72033_rp - Slot 5
!!!!!!!!!!!!

Activation will affect the following processes:
cdp2.iosproc

[output omitted]

Computing and verifying file checksums
!!!!!!!!!!!!
[DONE]
6500#
```

[3] Requires a Cisco.com login.

4. Before activating the patch verify that it is in a pending install state by using the show install running command. The patch can be activated by using the install activate <systemlocation> command. Note that the system will point out what processes need to be restarted. After a confirmation, the process will be restarted and the changes to the subsystem will become active in the system.

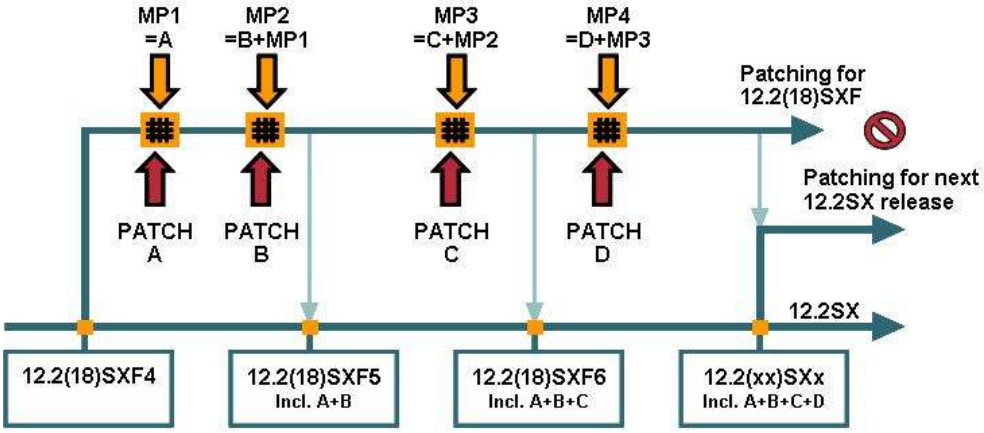
```
6500#show install running
Software running on card installed at location s72033_rp - Slot 5 :

B/P C State Filename
--- - -------- --------
B Active disk0:/sys/s72033_rp/base/DRACO2_MP
P PendInst disk0:/sys/s72033_rp/patch/patch-Yakhurana-00-0-n.so

Software running on card installed at location s72033 - Slot 5 :

B/P C State Filename
--- - -------- --------
B Active disk0:/sys/s72033/base/s72033-adventerprisek9_wan-vm
P PendInst disk0:/sys/s72033/patch/patch-Yakhurana-00-0-n.so

[output omitted]

6500#install activate disk0:/sys
Determining processes to restart on slot s72033_rp - Slot 5
!!!!!!!!!!

The following processes will be restarted:
cdp2.iosproc

[output omitted]

Do you want to continue with activating this change set...? [yes/no]: yes
Proceeding with activation ...
Affected processes restarted.

[DONE]

6500#

6500#show install running
Software running on card installed at location s72033_rp - Slot 5 :

B/P C State Filename
--- - ------ --------
B Active disk0:/sys/s72033_rp/base/DRACO2_MP
P Active disk0:/sys/s72033_rp/patch/patch-Yakhurana-00-0-n.so
```

```
Software running on card installed at location s72033 - Slot 5 :

B/P C State Filename
--- - ------ --------
B Active disk0:/sys/s72033/base/s72033-adventerprisek9_wan-vz
P Active disk0:/sys/s72033/patch/patch-Yakhurana-00-0-n.so

[output omitted]

6500#
```

Release Structure of Maintenance Packs

Initially Cisco will offer patches in the form of Maintenance Packs (MP), while each Maintenance Pack can contain one or multiple patches. Patches depend on a base image. Therefore when a Cisco IOS Software release is published, a Patch Integration Branch (PAIB) is started. In Figure 3, the red arrows indicate the point in time when a software defect is identified. Shortly after that, a Maintenance Pack (MP) is being made available on Cisco.com. Maintenance Packs will include all patches from previous Maintenance Packs offered for that specific PAIB.

Figure 3. Maintenance Pack Release Structure



As soon as a defect has been resolved and a Maintenance Pack has been released, the fix will also be synched to the base software train (this is the same as for previously delivered Cisco IOS Software releases). This assures that all defects known are immediately addressed and will be automatically integrated in the next maintenance release.

When maintenance support on a software release like 12.2(18)SXF ends, the PAIB will continue for at least one more month after which no more Maintenance Packs will be released. As soon as a new feature release is made available with the Cisco IOS Software Modularity enhancement, a new PAIB is initiated. All previously known patches from earlier PAIBs will be integrated in that new release. The normal release process and maintenance rebuild process is unaffected. The PAIB support is simply an extension to normal Cisco software support policies and can be compared with maintenance releases offering a more granular level.

INSTALLER

To provide the capability of managing patches, a new component has been added to the Cisco IOS Software infrastructure-the installer. The installer offers not only adding and removing of Maintenance Packs but also setting tags and repackaging

images. It keeps track of the whole patching history of a system, be it the running system (where the system is bound to) or another system that is installed on the same media with a different base directory.

Checks

When a Maintenance Pack is added to the system, the system performs multiple checks:

- Do all parts of the Maintenance Pack match with the base image on the system?
- Has the Maintenance Pack already been installed on this system?
- Has the Maintenance Pack been installed before and pruned? (Pruning saves space by removing previously defined tags and files necessary for rollbacks on older patches.)
- Is there enough space on the file system to add the Maintenance Pack?
- Does the checksum match with the one stored in the Maintenance Pack?

The system performs further checks to assure all symbols from the system as well as the Maintenance Pack can be resolved.

The benefit of these steps is that only applicable Maintenance Packs can be installed. If a Maintenance Pack is corrupted during download or installation, the installer prevents the Maintenance Pack from being applied to the system.

Tags

A tag can be looked at as a label describing a single point in time of the patching history. The network administrator can define tags at any time. When a tag is defined and subsequently patches are added and activated it is not necessary to roll back patch by patch; the network administrator can return to the point in the patching history where the tag was defined using a single command. To ease operation there are three predefined tags which are provided by default (Table 2):

Table 2. Predefined Tags and Associated Actions

Tag name	Action
CISCO_BASE	Removes all patches applied to the base image
CISCO_LATEST	Rolls back the latest added patch[4]
CISCO_LATEST_ACTIVATE	Rolls back to the level of the previous “install activate”

To check the current state of the system the show install running command can be used (see previous section). To set a new tag the install commit command is used. It conserves the current information and associates it with the name given.

Rollback

While adding patches to a system is helpful, full patching capability also demands removal of patches during runtime. This process is called rollback. In case a patch does not include the change one had hoped for, removal must be possible to get the system back to the previous state.

To simplify patching, removing patches is only possible in the reverse order they were applied: if patch A, B, and C have been applied in that order to the system, the only way to roll back is removing patch C first, then B and lastly patch A. However, it is possible to remove multiple patches in one step. With this there is no possibility to violate any dependencies patches might have amongst each other. In any case, because patches are offered through Maintenance Packs, rollback is only allowed to a previous Maintenance Pack level.

[4] If more than one patch has been installed before the last “install activate” was performed, only the last installed patch will be removed.

Similar to adding patches, rollback is a two-step process. First a single patch or multiple patches that should be rolled back have to be “selected.” This is done with the help of tags using the install rollback <system location> <tag> command. The tag field refers either to a self-defined or predefined tag. In a second step these changes also need to be activated with the install activate command. The system will again list what kind of processes will need to be restarted and ask for a confirmation.

The system allows adding and removing multiple patches before the activation. Assume a system with a patch that needs to be removed while another patch should be installed (both patches would affect the same subsystem). As a first step the already active patch will be rolled back and the new patch is installed. As a second step all these changes are activated at once, and therefore only a single process restart is needed to accomplish rolling back and installing a new patch.

Repackaging

Over time a number of patches can be added to a base image. As described earlier, tags can be set, defining individual stages of the system. Repackaging helps deployment of images offering Cisco IOS Software Modularity on a large number of devices. The process of repackaging compresses the following information into one single binary file:

- Base image
- All Maintenance Packs that have been either installed or installed and activated on the system
- All tags

So not only files but also all of the patching history will be transformed into a single file.

A possible scenario is that in a lab/test bed environment, a base image with necessary patches is loaded. After the release has been qualified to suffice the needs of the production network, a repackage of the system can be performed. The repackaged image will be published on an internal TFTP server and all new systems will obtain their image from a central “master” image. All patching information is available in the exact same detail as on the “original” image. When the system is running from this installed image, adding and rolling back patches can be performed as if the base image had been on this system from the beginning. Repackaged images must be installed onto a system and cannot be booted as a single binary file.

Pruning

Because of the addition of multiple patches to the system while keeping the full history and rollback possibility, the file system can grow in size. To optimize the size of the file system, the installer offers the option to “prune.” Pruning can only be performed for patches that have been superseded by newer patches. Pruning removes previously defined tags and files necessary for rollbacks. While this saves space from a file system point of view, one needs to note that this removes the option to roll back to previous stages in the patching history.

A possible application can be seen in the case that a base image has been enhanced with some patches and now is qualified as a “new base” for the corporate network. Because in this case no rollbacks will be needed anymore, a system can be pruned before it is repackaged for distribution in the network. This would save some more space on the internal file system of the individual Cisco Catalyst 6500 Series switches.

EMBEDDED EVENT MANAGER

The Embedded Event Manager (EEM) represents another powerful enhancement to the Cisco IOS Software infrastructure that gets introduced at the same time as Software Modularity. Based on defined events, custom actions can be executed on the local system. Triggers for actions as well as the individual steps themselves can be defined with the help of Tool Command Language (TCL) scripts. This gives the user the freedom of customizing the trigger as well as the action taken according to the individual needs.

Since EEM is part of the Cisco IOS Software Infrastructure it can act autonomously even if connectivity to a central management station is temporarily unavailable. To describe the architecture of EEM it can be broken down into three components:

- Event Detectors
- Policy Engine
- Embedded Event Manager Server

Event Detectors can be viewed as agents in various parts the operating system. These Event Detectors can then trigger the execution of scripts which contains custom actions. Event Detectors can publish events[5] based upon CLI input, counters, resource thresholds, timer based services, SNMP and SYSLOG messages, routing protocol events and more. For a complete list of Event Detectors please consult the EEM documentation.

The Policy Engine binds the user-defined policies to the system. The policy engine offers two interfaces to do so:

- TCL scripts
- CLI applets

The policy engine for Tool Command Language (TCL) scripts offers a TCL interface. While some predefined scripts are part of the system, network operators can make use of the TCL scripting interface to add their own scripts to have the system perform actions based on the individual need. Actions that can be performed reach from gathering output from specifiable commands up to full patch management for switches.

Embedded Event Manager Server finally ties all this together. The event detectors deliver their output to the EEM server where the server makes use of the policy engine.

A simple example could make use of the SYSLOG event detector. As soon as a SYSLOG message indicating an OSPF adjacency change "%OSPF-5-ADJCHG" is issued by the system, the output of the command "show ip ospf neighbor detail" could be stored on a local file system.

Having both Software Modularity and EEM provides an even more powerful combination. EEM, which itself is a modularized process, can take action based on process behavior. For example, in the event of a process crash, relevant information such as crash dump and memory allocation information can be stored locally or on a centralized server to ease troubleshooting. Once this is done, the switch can send a notification to the network administrator, who in return can then do further root cause analysis or contact the Cisco Technical Assistance Center (TAC).

To react on specific system behavior using user-defined scripts greatly strengthens the ability to include the Cisco Catalyst 6500 Series into a variety networks. Be it from a small and medium business customer up to the largest networks today.

APPENDIX

Migrating to a Release Offering Software Modularity

Before migrating a system, check the release notes for hardware and software compatibility (including ROMMON version). The following steps are necessary to migrate a system that is running in native Cisco IOS Software mode[6].

[5] "Publish" is another term for notifying the EEM server that an event has occurred.
[6] A system running a single binary file for switch processor and route processor not offering Software Modularity.

Download the appropriate image from the Software Center[7] on Cisco.com. Images offering Software Modularity are delivered as a single binary file and can be identified as such based on the "-vz" in the file name.

1. Make sure that you have enough space on either bootdisk:, disk0: or disk1: and copy the image to the system. If you want to install the image make sure that you have about 2.5 times as much space as the size of the binary file[8].
2. Change the boot variable to match the new image using the boot system flash... command.
3. Save the configuration, verify that the boot variable matches the image offering Software Modularity using the show bootvar command and reload the system.
4. Issue the show version command to verify that new image has been loaded.

To run the system with an "installed image," the procedure listed in the "Modes of operation and installing an image" section of this document should be followed.

REFERENCES AND CONTACTS

Cisco IOS Software Modularity for the Catalyst 6500 Product Bulletin (contains details on hardware support) http://www.cisco.com/go/6500swmod

Cisco Catalyst 6500 Release Notes: http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/relnotes/index.htm

Cisco Catalyst 6500 Documentation: http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/index.htm or http://www.cisco.com/go/catalyst6500

Cisco IOS Software Modularity Patch Navigator: http://www.cisco.com/go/pn

LAN Switching Software Download: (Cisco.com login is required to view this content)

http://www.cisco.com/kobayashi/sw-center/lan/cat6000.shtml

For additional information, please contact your local sales office.

[7] Requires a Cisco.com login.
[8] Assuming the network administrator wants to leave space for installing a second image and some patches.

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

C11-332597-00 02/06