Refining an IP Access List

There are several ways to refine an access list while or after you create it. You can change the order of the entries in an access list or add entries to an access list. You can restrict access list entries to a certain time of day or week, or achieve finer granularity when filtering packets by filtering noninitial fragments of packets.

Your software release may not support all the features that are documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. The Feature Information Table at the end of this document provides information about the documented features and lists the releases in which each feature is supported.

Contents

- Hardware Compatibility Matrix for the Cisco cBR Series Routers, on page 1
- Information About Refining an IP Access List, on page 3
- How to Refine an IP Access List, on page 6
- Configuration Examples for Refining an IP Access List, on page 11
- Additional References, on page 13
- Feature Information for Refining an IP Access List, on page 14

Hardware Compatibility Matrix for the Cisco cBR Series Routers

Note The hardware components that are introduced in a given Cisco IOS-XE Release are supported in all subsequent releases unless otherwise specified.

Table 1: Hardware Compatibility Matrix for the Cisco cBR Series Routers

Cisco CMTS Platform	Processor Engine	Interface Cards
Cisco cBR-8 Converged Broadband Router	**Cisco IOS-XE Release 16.5.1 and Later Releases** Cisco cBR-8 Supervisor: • PID—CBR-SUP-250G • PID—CBR-CCAP-SUP-160G	**Cisco IOS-XE Release 16.5.1 and Later Releases** Cisco cBR-8 CCAP Line Cards: • PID—CBR-LC-8D30-16U30 • PID—CBR-LC-8D31-16U30 • PID—CBR-RF-PIC • PID—CBR-RF-PROT-PIC • PID—CBR-CCAP-LC-40G • PID—CBR-CCAP-LC-40G-R • PID—CBR-CCAP-LC-G2-R • PID—CBR-SUP-8X10G-PIC • PID—CBR-2X100G-PIC Digital PICs: • PID—CBR-DPIC-8X10G • PID—CBR-DPIC-2X100G Cisco cBR-8 Downstream PHY Module: • PID—CBR-D31-DS-MOD Cisco cBR-8 Upstream PHY Modules: • PID—CBR-D31-US-MOD

Note Do not use DPICs (8X10G and 2x100G) to forward IP traffic, as it may cause buffer exhaustion, leading to line card reload.

The only allowed traffic on a DPIC interface is DEPI, UEPI, and GCP traffic from the Cisco cBR-8 router to Remote PHY devices. Other traffic such as DHCP, SSH, and UTSC should flow via another router, since DPICs cannot be used for normal routing.

Information About Refining an IP Access List

Access List Sequence Numbers

The ability to apply sequence numbers to IP access list entries simplifies access list changes. Prior to the IP Access List Entry Sequence Numbering feature, there was no way to specify the position of an entry within an access list. If you wanted to insert an entry in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

Sequence numbers allow users to add access list entries and resequence them. When you add a new entry, you specify the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry.

Benefits of Access List Sequence Numbers

An access list sequence number is a number at the beginning of a **permit** or **deny** command in an access list. The sequence number determines the order that the entry appears in the access list. The ability to apply sequence numbers to IP access list entries simplifies access list changes.

Prior to having sequence numbers, users could only add access list entries to the end of an access list; therefore, needing to add statements anywhere except the end of the list required reconfiguring the entire access list. There was no way to specify the position of an entry within an access list. If a user wanted to insert an entry (statement) in the middle of an existing list, all of the entries after the desired position had to be removed, then the new entry was added, and then all the removed entries had to be reentered. This method was cumbersome and error prone.

This feature allows users to add sequence numbers to access list entries and resequence them. When a user adds a new entry, the user chooses the sequence number so that it is in a desired position in the access list. If necessary, entries currently in the access list can be resequenced to create room to insert the new entry. Sequence numbers make revising an access list much easier.

Sequence Numbering Behavior

- For backward compatibility with previous releases, if entries with no sequence numbers are applied, the first entry is assigned a sequence number of 10, and successive entries are incremented by 10. The maximum sequence number is 2147483647. If the generated sequence number exceeds this maximum number, the following message is displayed:

```
Exceeded maximum sequence number.
```

- If the user enters an entry without a sequence number, it is assigned a sequence number that is 10 greater than the last sequence number in that access list and is placed at the end of the list.
- If the user enters an entry that matches an already existing entry (except for the sequence number), then no changes are made.
- If the user enters a sequence number that is already present, the following error message is generated:

```
Duplicate sequence number.
```

- If a new access list is entered from global configuration mode, then sequence numbers for that access list are generated automatically.
- Sequence numbers are not nvgened. That is, the sequence numbers themselves are not saved. In the event that the system is reloaded, the configured sequence numbers revert to the default sequence starting number and increment. The function is provided for backward compatibility with software releases that do not support sequence numbering.
- This feature works with named and numbered, standard and extended IP access lists.

Benefits of Time Ranges

Benefits and possible uses of time ranges include the following:

- The network administrator has more control over permitting or denying a user access to resources. These resources could be an application (identified by an IP address/mask pair and a port number), policy routing, or an on-demand link (identified as interesting traffic to the dialer).
- Network administrators can set time-based security policy, including the following:
  - Perimeter security using access lists
  - Data confidentiality with IP Security Protocol (IPsec)
- When provider access rates vary by time of day, it is possible to automatically reroute traffic cost effectively.
- Network administrators can control logging messages. Access list entries can log traffic at certain times of the day, but not constantly. Therefore, administrators can simply deny access without needing to analyze many logs generated during peak hours.

Benefits Filtering Noninitial Fragments of Packets

Filter noninitial fragments of packets with an extended access list if you want to block more of the traffic you intended to block, not just the initial fragment of such packets. You should first understand the following concepts.

If the **fragments**keyword is used in additional IP access list entries that deny fragments, the fragment control feature provides the following benefits:

Additional Security

You are able to block more of the traffic you intended to block, not just the initial fragment of such packets. The unwanted fragments no longer linger at the receiver until the reassembly timeout is reached because they are blocked before being sent to the receiver. Blocking a greater portion of unwanted traffic improves security and reduces the risk from potential hackers.

Reduced Cost

By blocking unwanted noninitial fragments of packets, you are not paying for traffic you intended to block.

Reduced Storage

By blocking unwanted noninitial fragments of packets from ever reaching the receiver, that destination does not have to store the fragments until the reassembly timeout period is reached.

Expected Behavior Is Achieved

The noninitial fragments will be handled in the same way as the initial fragment, which is what you would expect. There are fewer unexpected policy routing results and fewer fragments of packets being routed when they should not be.

Access List Processing of Fragments

The behavior of access list entries regarding the use or lack of use of the **fragments** keyword can be summarized as follows:

If the Access-List Entry Has...	Then...

...no **fragments** keyword (the default), and assuming all of the access-list entry information matches,	For an access list entry that contains only Layer 3 information:
	• The entry is applied to nonfragmented packets, initial fragments, and noninitial fragments.
	For an access list entry that contains Layer 3 and Layer 4 information:
	• The entry is applied to nonfragmented packets and initial fragments.
	  • If the entry is a **permit** statement, then the packet or fragment is permitted.
	  • If the entry is a **deny** statement, then the packet or fragment is denied.
	• The entry is also applied to noninitial fragments in the following manner. Because noninitial fragments contain only Layer 3 information, only the Layer 3 portion of an access list entry can be applied. If the Layer 3 portion of the access list entry matches, and
	  • If the entry is a **permit** statement, then the noninitial fragment is permitted.
	  • If the entry is a **deny** statement, then the next access list entry is processed.
	Note
	The **deny** statements are handled differently for noninitial fragments versus nonfragmented or initial fragments.

...the **fragments** keyword, and assuming all of the access-list entry information matches,	The access list entry is applied only to noninitial fragments. The **fragments** keyword cannot be configured for an access list entry that contains any Layer 4 information.

Be aware that you should not add the **fragments** keyword to every access list entry because the first fragment of the IP packet is considered a nonfragment and is treated independently of the subsequent fragments. An initial fragment will not match an access list **permit** or **deny** entry that contains the **fragments** keyword. The

packet is compared to the next access list entry, and so on, until it is either permitted or denied by an access list entry that does not contain the **fragments** keyword. Therefore, you may need two access list entries for every **deny** entry. The first **deny** entry of the pair will not include the **fragments** keyword and applies to the initial fragment. The second **deny** entry of the pair will include the **fragments** keyword and applies to the subsequent fragments. In the cases in which there are multiple **deny** entries for the same host but with different Layer 4 ports, a single **deny** access list entry with the **fragments** keyword for that host is all that needs to be added. Thus all the fragments of a packet are handled in the same manner by the access list.

Packet fragments of IP datagrams are considered individual packets, and each counts individually as a packet in access list accounting and access list violation counts.

How to Refine an IP Access List

The tasks in this module provide you with various ways to refine an access list if you did not already do so while you were creating it. You can change the order of the entries in an access list, add entries to an access list, restrict access list entries to a certain time of day or week, or achieve finer granularity when filtering packets by filtering on noninitial fragments of packets.

Revising an Access List Using Sequence Numbers

Perform this task if you want to add entries to an existing access list, change the order of entries, or simply number the entries in an access list to accommodate future changes.

Note Remember that if you want to delete an entry from an access list, you can simply use the **no deny** or **no permit** form of the command, or the **no** *sequence-number* command if the statement already has a sequence number.

Note

- Access list sequence numbers do not support dynamic, reflexive, or firewall access lists.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list resequence** *access-list-name starting-sequence-number increment*
4. **ip access-list** {**standard**| **extended**} *access-list-name*
5. Do one of the following:
   - *sequence-number* **permit** *source source-wildcard*
   - *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]
6. Do one of the following:
   - *sequence-number* **deny** *source source-wildcard*
   - *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**]

7. Repeat Step 5 and Step 6 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.
8. **end**
9. **show ip access-lists** *access-list-name*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list resequence** *access-list-name starting-sequence-number increment* **Example:** `Router(config)# ip access-list resequence kmd1 100 15`	Resequences the specified IP access list using the starting sequence number and the increment of sequence numbers. • This example resequences an access list named kmd1. The starting sequence number is 100 and the increment is 15.
Step 4	**ip access-list** {**standard**| **extended**} *access-list-name* **Example:** `Router(config)# ip access-list standard xyz123`	Specifies the IP access list by name and enters named access list configuration mode. • If you specify **standard**, make sure you specify subsequent **permit** and **deny** statements using the standard access list syntax. • If you specify **extended**, make sure you specify subsequent **permit** and **deny** statements using the extended access list syntax.
Step 5	Do one of the following: • *sequence-number* **permit** *source source-wildcard* • *sequence-number* **permit** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Router(config-std-nacl)# 105 permit 10.5.5.5 0.0.0.255`	Specifies a permit statement in named IP access list mode. • This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. • See the **permit** (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP). • Use the **no** *sequence-number* command to delete an entry.

	Command or Action	Purpose
		• As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Router(config-ext-nacl)# and you would use the extended **permit** command syntax.
Step 6	Do one of the following: • *sequence-number* **deny** *source source-wildcard* • *sequence-number* **deny** *protocol source source-wildcard destination destination-wildcard* [**precedence** *precedence*][**tos** *tos*] [**log**] [**time-range** *time-range-name*] [**fragments**] **Example:** `Router(config-std-nacl)# 110 deny 10.6.6.7 0.0.0.255`	(Optional) Specifies a deny statement in named IP access list mode. • This access list happens to use a **permit**statement first, but a **deny** statement could appear first, depending on the order of statements you need. • See the **deny** (IP) command for additional command syntax to permit upper layer protocols (ICMP, IGMP, TCP, and UDP). • Use the **no** *sequence-number* command to delete an entry. • As the prompt indicates, this access list was a standard access list. If you had specified **extended** in Step 4, the prompt for this step would be Router(config-ext-nacl)# and you would use the extended **deny** command syntax.
Step 7	Repeat Step 5 and Step 6 as necessary, adding statements by sequence number where you planned. Use the **no** *sequence-number* command to delete an entry.	Allows you to revise the access list.
Step 8	**end** **Example:** `Router(config-std-nacl)# end`	(Optional) Exits the configuration mode and returns to privileged EXEC mode.
Step 9	**show ip access-lists** *access-list-name* **Example:** `Router# show ip access-lists xyz123`	(Optional) Displays the contents of the IP access list. • Review the output to see that the access list includes the new entry.

Examples

The following is sample output from the **show ip access-lists** command when the **xyz123** access list is specified.

```
Router# show ip access-lists xyz123
Standard IP access list xyz123
100 permit 10.4.4.0, wildcard bits 0.0.0.255
105 permit 10.5.5.5, wildcard bits 0.0.0.255
115 permit 10.0.0.0, wildcard bits 0.0.0.255
```

```
130 permit 10.5.5.0, wildcard bits 0.0.0.255
145 permit 10.0.0.0, wildcard bits 0.0.0.255
```

Restricting an Access List Entry to a Time of Day or Week

By default, access list statements are always in effect once they are applied. However, you can define the times of the day or week that **permit** or **deny** statements are in effect by defining a time range, and then referencing the time range by name in an individual access list statement. IP and Internetwork Packet Exchange (IPX) named or numbered extended access lists can use time ranges.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip access-list extended** *name*
4. [*sequence-number*] **deny** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]]
5. [*sequence-number*] **deny** *protocol source*[*source-wildcard*][*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **fragments**
6. [*sequence-number*] **permit** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]]
7. Repeat some combination of Steps 4 through 6 until you have specified the values on which you want to base your access list.
8. **end**
9. **show ip access-list**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip access-list extended** *name* **Example:** `Router(config)# ip access-list extended rstrct4`	Defines an extended IP access list using a name and enters extended named access list configuration mode.

	Command or Action	Purpose
Step 4	[*sequence-number*] **deny** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **Example:** `Router(config-ext-nacl)# deny ip any 172.20.1.1`	(Optional) Denies any packet that matches all of the conditions specified in the statement. • This statement will apply to nonfragmented packets and initial fragments.
Step 5	[*sequence-number*] **deny** *protocol source*[*source-wildcard*][*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **fragments** **Example:** `Router(config-ext-nacl)# deny ip any 172.20.1.1 fragments`	(Optional) Denies any packet that matches all of the conditions specified in the statement • This statement will apply to noninitial fragments.
Step 6	[*sequence-number*] **permit** *protocol source*[*source-wildcard*] [*operator port*[*port*]] *destination*[*destination-wildcard*] [*operator port*[*port*]] **Example:** `Router(config-ext-nacl)# permit tcp any any`	Permits any packet that matches all of the conditions specified in the statement. • Every access list needs at least one **permit** statement. • If the *source-wildcard* or *destination-wildcard*isomitted, a wildcard mask of 0.0.0.0 is assumed, meaning match on all bits of the source or destination address, respectively. • Optionally use the keyword **any** as a substitute for the *source source-wildcard*or *destination destination-wildcard*to specify the address and wildcard of 0.0.0.0 255.255.255.255.
Step 7	Repeat some combination of Steps 4 through 6 until you have specified the values on which you want to base your access list.	Remember that all sources not specifically permitted are denied by an implicit **deny** statement at the end of the access list.
Step 8	**end** **Example:** `Router(config-ext-nacl)# end`	Ends configuration mode and returns the system to privileged EXEC mode.
Step 9	**show ip access-list** **Example:** `Router# show ip access-list`	(Optional) Displays the contents of all current IP access lists.

What to Do Next

Apply the access list to an interface or reference it from a command that accepts an access list.

Note To effectively eliminate all packets that contain IP Options, we recommend that you configure the global **ip options drop** command.

Configuration Examples for Refining an IP Access List

Example Resequencing Entries in an Access List

The following example shows an access list before and after resequencing. The starting value is 1, and increment value is 2. The subsequent entries are ordered based on the increment values that users provide, and the range is from 1 to 2147483647.

When an entry with no sequence number is entered, by default it has a sequence number of 10 more than the last entry in the access list.

```
Router# show access-list carls
Extended IP access list carls
    10 permit ip host 10.3.3.3 host 172.16.5.34
    20 permit icmp any any
    30 permit tcp any host 10.3.3.3
    40 permit ip host 10.4.4.4 any
    50 Dynamic test permit ip any any
    60 permit ip host 172.16.2.2 host 10.3.3.12
    70 permit ip host 10.3.3.3 any log
    80 permit tcp host 10.3.3.3 host 10.1.2.2
    90 permit ip host 10.3.3.3 any
    100 permit ip any any
Router(config)# ip access-list extended carls
Router(config)# ip access-list resequence carls 1 2
Router(config)# end
Router# show access-list carls
Extended IP access list carls
    1 permit ip host 10.3.3.3 host 172.16.5.34
    3 permit icmp any any
    5 permit tcp any host 10.3.3.3
    7 permit ip host 10.4.4.4 any
    9 Dynamic test permit ip any any
    11 permit ip host 172.16.2.2 host 10.3.3.12
    13 permit ip host 10.3.3.3 any log
    15 permit tcp host 10.3.3.3 host 10.1.2.2
    17 permit ip host 10.3.3.3 any
    19 permit ip any any
```

Example Adding an Entry with a Sequence Number

In the following example, an new entry (sequence number 15) is added to an access list:

```
Router# show ip access-list
Standard IP access list tryon
2 permit 10.4.4.2, wildcard bits 0.0.255.255
5 permit 10.0.0.44, wildcard bits 0.0.0.255
10 permit 10.0.0.1, wildcard bits 0.0.0.255
20 permit 10.0.0.2, wildcard bits 0.0.0.255
```

```
Router(config)# ip access-list standard tryon
Router(config-std-nacl)# 15 permit 10.5.5.5 0.0.0.255
Router# show ip access-list
Standard IP access list tryon
2 permit 10.4.0.0, wildcard bits 0.0.255.255
5 permit 10.0.0.0, wildcard bits 0.0.0.255
10 permit 10.0.0.0, wildcard bits 0.0.0.255
15 permit 10.5.5.0, wildcard bits 0.0.0.255
20 permit 10.0.0.0, wildcard bits 0.0.0.255
```

Example Adding an Entry with No Sequence Number

The following example shows how an entry with no specified sequence number is added to the end of an access list. When an entry is added without a sequence number, it is automatically given a sequence number that puts it at the end of the access list. Because the default increment is 10, the entry will have a sequence number 10 higher than the last entry in the existing access list.

```
Router(config)# ip access-list standard resources
Router(config-std-nacl)# permit 10.1.1.1 0.0.0.255
Router(config-std-nacl)# permit 10.2.2.2 0.0.0.255
Router(config-std-nacl)# permit 10.3.3.3 0.0.0.255
Router# show access-list
Standard IP access list resources
10 permit 10.1.1.1, wildcard bits 0.0.0.255
20 permit 10.2.2.2, wildcard bits 0.0.0.255
30 permit 10.3.3.3, wildcard bits 0.0.0.255
Router(config)# ip access-list standard resources
Router(config-std-nacl)# permit 10.4.4.4 0.0.0.255
Router(config-std-nacl)# end
Router# show access-list
Standard IP access list resources
10 permit 10.1.1.1, wildcard bits 0.0.0.255
20 permit 10.2.2.2, wildcard bits 0.0.0.255
30 permit 10.3.3.3, wildcard bits 0.0.0.255
40 permit 10.4.4.4, wildcard bits 0.0.0.255
```

Example Time Ranges Applied to IP Access List Entries

The following example creates a time range called no-http, which extends from Monday to Friday from 8:00 a.m. to 6:00 p.m. That time range is applied to the **deny** statement, thereby denying HTTP traffic on Monday through Friday from 8:00 a.m. to 6:00 p.m.

The time range called udp-yes defines weekends from noon to 8:00 p.m. That time range is applied to the **permit** statement, thereby allowing UDP traffic on Saturday and Sunday from noon to 8:00 p.m. only. The access list containing both statements is applied to inbound packets on Ten Gigabit Ethernet interface 4/1/0.

```
time-range no-http
 periodic weekdays 8:00 to 18:00
!
time-range udp-yes
 periodic weekend 12:00 to 20:00
!
ip access-list extended strict
 deny tcp any any eq http time-range no-http
 permit udp any any time-range udp-yes
!
```

```
interface TenGigabitEthernet4/1/0
 ip access-group strict in
```

Example Filtering IP Packet Fragments

In the following access list, the first statement will deny only noninitial fragments destined for host 172.16.1.1. The second statement will permit only the remaining nonfragmented and initial fragments that are destined for host 172.16.1.1 TCP port 80. The third statement will deny all other traffic. In order to block noninitial fragments for any TCP port, we must block noninitial fragments for all TCP ports, including port 80 for host 172.16.1.1. That is, non-initial fragments will not contain Layer 4 port information, so, in order to block such traffic for a given port, we have to block fragments for all ports.

```
access-list 101 deny ip any host 172.16.1.1 fragments
access-list 101 permit tcp any host 172.16.1.1 eq 80
access-list 101 deny ip any any
```

Additional References

Related Documents

Related Topic	Document Title
Using the **time-range** command to establish time ranges	The chapter *Performing Basic System Management* in the *Cisco IOS XE Network Management Configuration Guide*
Network management command descriptions	*Cisco IOS Network Management Command Reference*

Standards

Standard	Title
No new or modified standards are supported by this feature, and support for existing standards has not been modified by this feature.	--

MIBs

MIB	MIBs Link
No new or modified MIBs are supported by this feature, and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE software releases, and feature sets, use Cisco MIB Locator found at the following URL: https://cfnng.cisco.com/mibs

RFCs

RFC	Title
No new or modified RFCs are supported by this feature, and support for existing RFCs has not been modified by this feature.	--

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Refining an IP Access List

Note The following table lists the software release in which a given feature is introduced. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 2: Feature Information for Refining an IP Access List

Feature Name	Releases	Feature Information
IP access lists	Cisco IOS XE Fuji 16.7.1	This feature was integrated into Cisco IOS XE Fuji 16.7.1 on theCisco cBR Series Converged Broadband Routers.