CCNP Routing and Switching
Portable Command Guide

All the CCNP ROUTE 300-101 and SWITCH 300-115 commands in one compact, portable resource

Scott Empson
Patrick Gargano
Hans Roth

ciscopress.com

CCNP Routing and Switching Portable Command Guide

Scott Empson
Patrick Gargano
Hans Roth

Cisco Press
800 East 96th Street
Indianapolis, Indiana 46240 USA

CCNP Routing and Switching Portable Command Guide

Scott Empson, Patrick Gargano, Hans Roth

Published by:

Cisco Press

800 East 96th Street

Indianapolis, IN 46240 USA

Printed in the United States of America

First Printing December 2014

Library of Congress Control Number: 2014955978

ISBN-13: 978-1-58714-434-9

ISBN-10: 1-58714-434-4

Warning and Disclaimer

This book is designed to provide information about the CCNP Route (300-101) and CCNP SWITCH (300-115) exams. Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied.

The information is provided on an "as is" basis. The authors, Cisco Press, and Cisco Systems, Inc. shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the discs or programs that may accompany it.

The opinions expressed in this book belong to the author and are not necessarily those of Cisco Systems, Inc.

Trademark Acknowledgments

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Cisco Press or Cisco Systems, Inc., cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact international@pearsoned.com.

Feedback Information

At Cisco Press, our goal is to create in-depth technical books of the highest quality and value. Each book is crafted with care and precision, undergoing rigorous development that involves the unique expertise of members from the professional technical community.

Readers' feedback is a natural continuation of this process. If you have any comments regarding how we could improve the quality of this book, or otherwise alter it to better suit your needs, you can contact us through email at feedback@ciscopress.com. Please make sure to include the book title and ISBN in your message.

We greatly appreciate your assistance.

Publisher
Paul Boger

Associate Publisher
Dave Dusthimer

Business Operation Manager, Cisco Press
Jan Cornelssen

Executive Editor
Mary Beth Ray

Managing Editor
Sandra Schroeder

Senior Development Editor
Christopher Cleveland

Senior Project Editor
Tonya Simpson

Copy Editor
Keith Cline

Technical Editor
Diane Teare

Editorial Assistant
Vanessa Evans

Cover Designer
Mark Shirar

Composition
Tricia Bronkella

Proofreader
Jess DeGabriele

Contents at a Glance

Table of Contents

Part II: SWITCH

About the Authors

Scott Empson is the chair of the Bachelor of Applied Information Systems Technology degree program at the Northern Alberta Institute of Technology in Edmonton, Alberta, Canada, where he teaches Cisco routing, switching, network design, and leadership courses in a variety of different programs (certificate, diploma, and applied degree) at the postsecondary level. Scott is also the program coordinator of the Cisco Networking Academy Program at NAIT, an area support center for the province of Alberta. He has a Masters of Education degree along with three undergraduate degrees: a Bachelor of Arts, with a major in English; a Bachelor of Education, again with a major in English/Language Arts; and a Bachelor of Applied Information Systems Technology, with a major in Network Management. He currently holds several industry certifications, including CCNP, CCDP, CCAI, ClEH, and Network+. Before instructing at NAIT, he was a junior/senior high school English/Language Arts/Computer Science teacher at different schools throughout Northern Alberta. Scott lives in Edmonton, Alberta, with his wife, Trina, and two children, Zach and Shae.

Patrick Gargano has been a Cisco Networking Academy Instructor since 2000. He currently heads the Networking Academy program and teaches CCNA/CCNP-level courses at Collège La Cité in Ottawa, Canada, where he has successfully introduced mastery-based learning and gamification into his teaching. In 2013 and 2014, Patrick led the Cisco Networking Academy student "Dream Team," which deployed the wired and wireless networks for attendees of the Cisco Live conferences in the United States. In 2014, Collège La Cité awarded him the prize for innovation and excellence in teaching. Previously he was a Cisco Networking Academy instructor at Cégep de l'Outaouais (Gatineau, Canada) and Louis-Riel High School (Ottawa, Canada) and a Cisco instructor (CCSI) for Fast Lane UK (London). His certifications include CCNA (R&S), CCNA Wireless, CCNA Security, and CCNP (R&S). #CiscoChampion @PatrickGargano

Hans Roth is an instructor in the Electrical Engineering Technology department at Red River College in Winnipeg, Manitoba, Canada. Hans has been teaching at the college for 17 years and teaches in both the engineering technology and IT areas. He has been with the Cisco Networking Academy since 2000, teaching CCNP curricula. Before teaching, Hans spent 15 years on R&D/product development teams helping design microcontroller-based control systems for consumer products and for the automotive and agricultural industries.

About the Technical Reviewer

Diane Teare, P.Eng, CCNP, CCDP, CCSI, PMP, is a professional in the networking, training, project management, and e-learning fields. She has more than 25 years of experience in designing, implementing, and troubleshooting network hardware and software and has been involved in teaching, course design, and project management. She has extensive knowledge of network design and routing technologies. Diane is a Cisco Certified Systems Instructor (CCSI) and holds her Cisco Certified Network Professional (CCNP), Cisco Certified Design Professional (CCDP), and Project Management Professional (PMP) certifications. She is an instructor, and the course director for the CCNA and CCNP Routing and Switching curriculum with one of the largest authorized Cisco Learning Partners. She was the director of e-learning for the same company, where she was responsible for planning and supporting all of the company's e-learning offerings in Canada, including Cisco courses. Diane has a bachelor's degree in applied science in electrical engineering and a master's degree in applied science in management science. Diane has authored, co-authored, and served as a technical reviewer on multiple Cisco Press titles.

Dedications

As always, this book is dedicated to Trina, Zach, and Shae. —Scott Empson

To my wife, Kathryn, for her patience, encouragement, love and understanding. I am a much better person thanks to her (or so she says. She also says there should be a comma after "love."). —Patrick Gargano

I'd like to again thank my wife, Carol, for her constant support and understanding during those times I've spent writing in the basement. —Hans Roth

Acknowledgments

Anyone who has ever had anything to do with the publishing industry knows that it takes many, many people to create a book. Our names may be on the cover, but there is no way that we can take credit for all that occurred to get this book from idea to publication. Therefore, we must thank the following:

Scott: The team at Cisco Press. Once again, you amaze me with your professionalism and the ability to make me look good. Mary Beth, Chris, and Tonya—thank you for your continued support and belief in my little engineering journal.

To my technical reviewer, Diane Teare, thanks for keeping me on track and making sure that what I wrote was correct and relevant. I have read and used Diane's books for many years, and now I finally have a chance to work with you. Hopefully, I live up to your standards.

A big thank you goes to my co-authors, Hans Roth and Patrick Gargano, for helping me through this with all of your technical expertise and willingness to assist in trying to make my ideas a reality. I am truly honored to have you as part of the Portable Command Guide family.

Patrick: I feel I must also echo some of Scott's acknowledgments. As the "new guy" on the team, I would have been lost had it not been for Mary Beth's advice, Vanessa's patience, Chris' direction, and Diane's eagle eyes. Thank you for making me feel part of the gang. As well, massive thanks to Scott for bringing me on board for this revision of the *CCNP Portable Command Guide*. It was a pleasure working with him and Hans on this project. I hope I've managed to uphold the level of excellence these books have achieved over the years.

Hans: The overall effort is large and the involvement is wide to get any book completed. Working with you folks at Cisco Press has again been a wonderful partnership. Your ongoing professionalism, understanding, and patience have consistently helped me to do a little better each time I sit down to write.

To our technical reviewer, Diane Teare: Wow, thanks for making me go deep.

Scott and Patrick: Thanks for your help, positive approach, and expertise. It was a very great pleasure.

Command Syntax Conventions

The conventions used to present command syntax in this book are the same conventions used in the IOS Command Reference. The Command Reference describes these conventions as follows:

- **Boldface** indicates commands and keywords that are entered literally as shown. In actual configuration examples and output (not general command syntax), boldface indicates commands that are manually input by the user (such as a **show** command).
- *Italic* indicates arguments for which you supply actual values.
- Vertical bars (|) separate alternative, mutually exclusive elements.
- Square brackets ([ ]) indicate an optional element.
- Braces ({ }) indicate a required choice.
- Braces within brackets ([{ }]) indicate a required choice within an optional element.

Introduction

Welcome to *CCNP Routing and Switching Portable Command Guide*! This book is the result of a redesign by Cisco of their professional-level certification exams to more closely align with the industry's need for networking talent as we enter the era of "the Internet of Everything." The previous success of the last editions of both the ROUTE and SWITCH books prompted Cisco Press to approach me with a request to update the book with the necessary new content to help both students and IT professionals in the field study and prepare for the new CCNP ROUTE and SWITCH exams. This time around, after many long talks with Hans and Patrick, Cisco Press, and other trusted IT colleagues, the decision was made to combine both ROUTE and SWITCH into a single volume. Hopefully, you will find value in having both exams' content in a single (albeit slightly thicker) volume. For someone who originally thought that a Portable Command Guide would be fewer than 100 pages in length and limited to the Cisco Academy program for its complete audience, I am continually amazed that my little engineering journal has caught on with such a wide range of people throughout the IT community.

For those of you who have worked with these books before, thank you for looking at this one. I hope that it will help you as you prepare for the vendor exam, or assist you in your daily activities as a Cisco network administrator/manager. For those of you new to the Portable Command Guides, you are reading what is essentially a cleaned-up version of my own personal engineering journals—a small notebook that I carry around with me that contains little nuggets of information; commands that I use but then forget; IP address schemes for the parts of the network I work with only on occasion; and those little reminders for those concepts that you only work with once or twice a year, but still need to know when those times roll around. As an educator who teaches these topics to post-secondary students, the classes I teach sometime occur only once a year; all of you out there can attest to the fact that it is extremely difficult to remember all those commands all the time. Having a journal of commands at your fingertips, without having to search the Cisco website (or if the network is down and you are the one responsible for getting it back online) can be a real timesaver.

With the creation of the new CCNP exam objectives, there is always something new to read, or a new podcast to listen to, or another slideshow from Cisco Live that you missed or that you just want to review again. The engineering journal can be that central repository of information that will not weigh you down as you carry it from the office or cubicle to the server and infrastructure rooms in some remote part of the building or some branch office.

To make this guide a more realistic one for you to use, the folks at Cisco Press have decided to continue with an appendix of blank pages—pages that are there for you to put your own personal touches (your own configurations, commands that are not in this book but are needed in your world, and so on). That way, this book will hopefully look less like the authors' journals and more like your own.

Who Should Read This Book?

This book is for those people preparing for the CCNP ROUTE and/or SWITCH exams, whether through self-study, on-the-job training and practice, study within the Cisco Academy Program, or study through the use of a Cisco Training Partner. There are also some handy hints and tips along the way to make life a bit easier for you in this endeavor. It is small enough that you will find it easy to carry around with you. Big, heavy textbooks might look impressive on your bookshelf in your office, but can you really carry them all around with you when you are working in some server room or equipment closet somewhere?

Strategies for Exam Preparation

The strategy you use for CCNP ROUTE and SWITCH might differ slightly from strategies used by other readers, mainly based on the skills, knowledge, and experience you already have obtained. For instance, if you have attended a ROUTE or SWITCH course, you might take a different approach than someone who learned routing via on-the-job training. Regardless of the strategy you use or the background you have, this book is designed to help you get to the point where you can pass the exam with the least amount of time required. For instance, there is no need for you to practice or read about EIGRP, OSPF, HSRP, or VLANs if you fully understand it already. However, many people like to make sure that they truly know a topic and therefore read over material that they already know. Several book features will help you gain the confidence that you need to be convinced that you know some material already, and to also help you know what topics you need to study more.

How This Book Is Organized

Although this book could be read cover to cover, I strongly advise against it, unless you really are having problems sleeping at night. The book is designed to be a simple listing of those commands needed to be understood to pass the ROUTE and SWITCH exams. Portable Command Guides contain very little theory; it has been designed to list out commands needed at this level of study.

This book follows the list of objectives for the CCNP ROUTE and SWITCH exams:

Part I: ROUTE

- **Chapter 1, "Basic Networking and Routing Concepts"**: This chapter shows the Cisco Hierarchical Model of Network Design; the Cisco Enterprise Composite Network Model; static and default Routes; Administrative Distances; IPv6 Addresses; and RIPng.
- **Chapter 2, "EIGRP Implementation"**: This chapter deals with EIGRP—the design, implementation, verification, and troubleshooting of this protocol in both IPv4 and IPv6.
- **Chapter 3, "Implementing a Scalable Multiarea Network OSPF Based Solution"**: This chapter deals with OSPF; a review of configuring OSPF, both

single area (as a review) and multiarea. Topics again include the design, implementation, verification, and troubleshooting of the protocol in both IPv4 and IPv6.

- **Chapter 4, "Configuration of Redistribution"**: This chapter show how to manipulate routing information. Topics include prefix lists, distribution lists, route maps, route redistribution, and static routes in both IPv4 and IPv6.
- **Chapter 5, "Path Control Implementation"**: This chapter deals those tools and commands that you can use to help evaluate network performance issues and control the path. Topics include CEF, Cisco IOS IP SLAs, and policy-based routing using route maps in both IPv4 and IPv6.
- **Chapter 6, "Enterprise Internet Connectivity"**: This chapter starts with DHCP and NAT and then deals with the use of BGP to connect an enterprise network to a service provider. Topics include the configuration, verificiation, and troubleshooting of a BGP-based solution, BGP attributes, regular expressions, and BGP route filtering using access lists.
- **Chapter 7, "Routers and Router Protocol Hardening"**: This chapter starts with checklists to follow when securing Cisco routers and the components of a router security policy. It then moves into topics such as password encryption, SSH, secure SNMP, backups, logging, and Network Time Protocol (NTP), and finishes with authentication of EIGRP, OSPF, and BGP.

Part II: SWITCH

- **Chapter 8, "Basic Concepts and Network Design"**: This chapter covers topics such as SDM templates, LLDP, PoE, and switch verification commands.
- **Chapter 9, "Campus Network Architecture"**: This chapter provides information on virtual LANs—creating, verifying, and troubleshooting them, along with EtherChannel, DHCPv4 and DHCPv6, and configuring and verifying voice VLANs.
- **Chapter 10, "Implementing Spanning Tree"**: This chapter provides information on the configuration of spanning tree, along with commands used to verify the protocol and to configure enhancements to spanning tree, such as Rapid Spanning Tree and Multiple Spanning Tree. The Cisco STP Toolkit is also shown here, along with FlexLinks.
- **Chapter 11, "Implementing Inter-VLAN Routing"**: This chapter shows the different ways to enable inter-VLAN communication—using an external router or using SVIs on a multilayer switch.
- **Chapter 12, "Implementing High-Availability Networks"**: This chapter covers topics such as IP service level agreements, port mirroring, and switch virtualization.
- **Chapter 13, "First-Hop Redundancy Implementation"**: This chapter provides information needed to ensure that you have first-hop redundancy; HSRP, VRRP, and GLBP are shown here in both IPv4 and IPv6.
- **Chapter 14, "Campus Network Security"**: Security is the focus of this chapter. Topics covered include switch security recommended practices, static MAC addresses, port security, 802.1x authentication, mitigating VLAN hopping, DHCP snooping, DAI, and private VLANs.

This page intentionally left blank

CHAPTER 1

Basic Network and Routing Concepts

This chapter provides information about the following topics:

- Cisco Hierarchical Network Model
- Cisco Enterprise Composite Network Model
- Typically used routing protocols
- IGP versus EGP routing protocols
- Routing protocol comparison
- Administrative distances
- Static routes: **permanent** keyword
- Floating static routes
- Static routes and recursive lookups
- Default routes
- Verifying static routes
- Applying IPv6 addresses to interfaces
- Implementing RIP next generation (RIPng)
- Verifying and troubleshooting RIPng
- Configuration example: RIPng
- IPv6 ping
- IPv6 traceroute

Cisco Hierarchical Network Model

Figure 1-1 shows the Cisco Hierarchical Network Model.

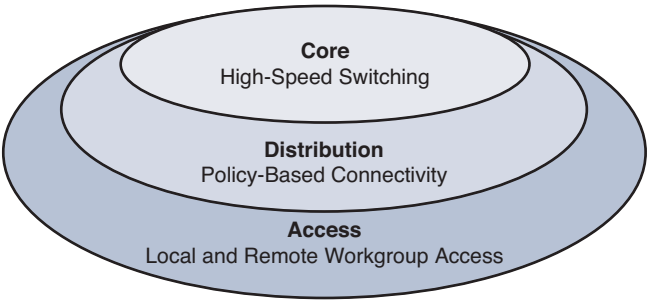


Figure 1-1 Cisco Hierarchical Network Model

Cisco Enterprise Composite Network Model

Figure 1-2 shows the Cisco Enterprise Composite Network Model.

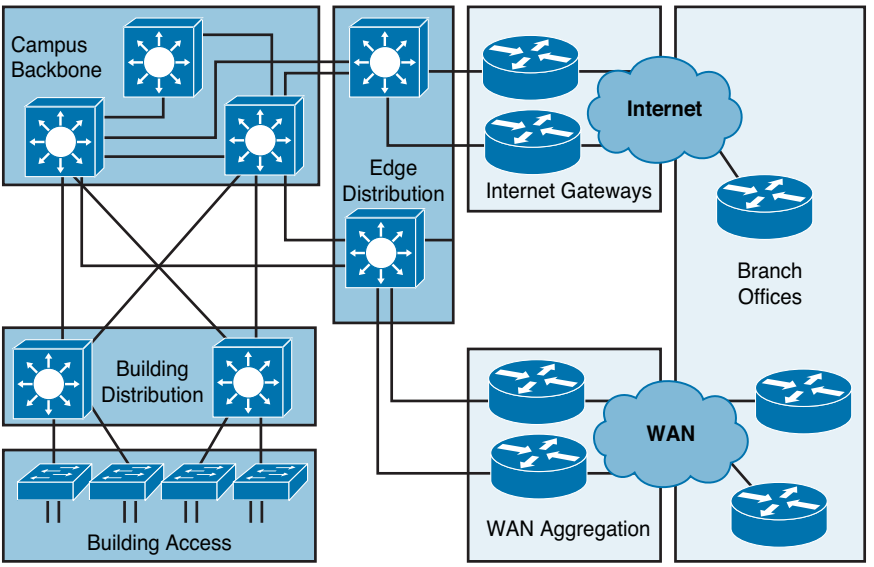


Figure 1-2 Cisco Enterprise Composite Network Model

Typically Used Routing Protocols

Figure 1-3 shows the most commonly used routing protocols.

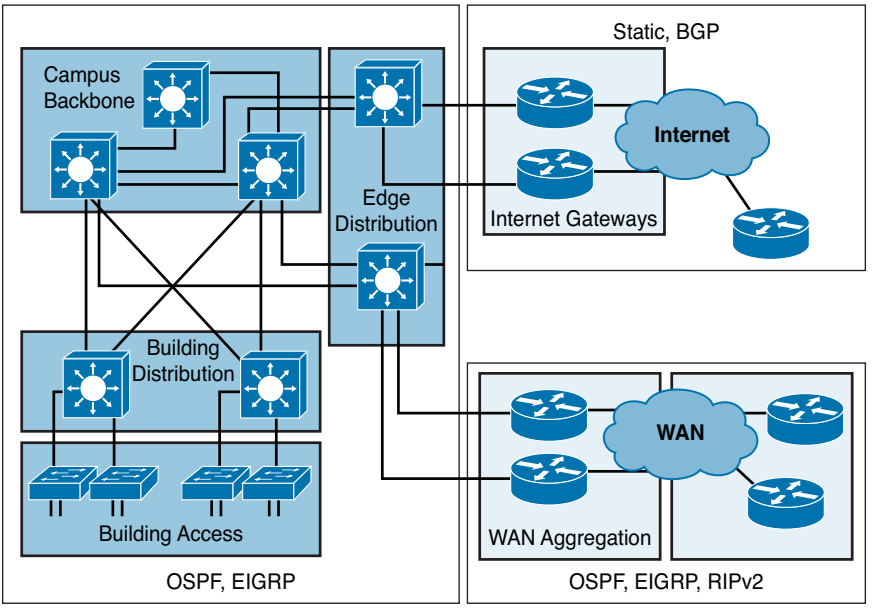


Figure 1-3 Typically Used Routing Protocols

IGP Versus EGP Routing Protocols

Figure 1-4 shows the location of IGP and EGP routing protocols.

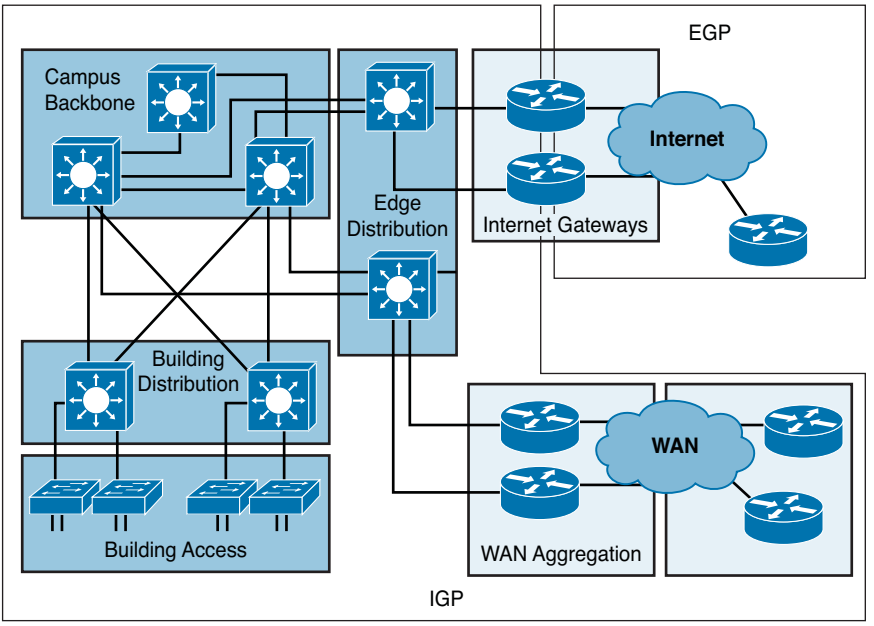


Figure 1-4 IGP Versus EGP Routing Protocols

Routing Protocol Comparison

The following table shows a comparison of Enhanced Interior Gateway Routing Protocol (EIGRP), Open Shortest Path First (OSPF) Protocol, and Border Gateway Protocol (BGP).

Parameters	EIGRP	OSPF	BGP
Size of network (small, medium, large, very large)	Large	Large	Very large
Speed of convergence (very high, high, medium, low)	Very high	High	Slow
Use of VLSM (yes, no)	Yes	Yes	Yes
Mixed-vendor devices (yes, no)	No	Yes	Yes

Administrative Distance

The Cisco default administrative distances (AD) are as follows.

Route Source	AD
Connected interface	0
Static route	1
EIGRP summary route	5

Route Source	AD
External Border Gateway Protocol (eBGP)	20
Internal EIGRP	90
Interior Gateway Routing Protocol (IGRP) (no longer supported)	100
OSPF	110
Intermediate System-to-Intermediate System (IS-IS) Protocol	115
RIP	120
Exterior Gateway Protocol (EGP)	140
External EIGRP	170
Internal BGP (iBGP)	200
Unknown	255

The commands to change the AD of an OSPF route from its default setting are as follows. You use these same commands when changing the ADs for other protocols as well.

`Router(config)#router ospf 1`	Starts the OSPF routing process.
`Router(config-router)#distance 95`	Changes the AD of OSPF from 110 to 95.
`Router(config-router)#distance 105 192.168.10.2 0.0.0.0`	Applies an AD of 105 to all OSPF routes received from 192.169.10.2.
	NOTE: This newly assigned AD is locally significant only. All other routers will still apply an AD of 110 to these routes.
`Router(config-router)#distance 102 172.16.10.2 0.0.0.0`	Applies an AD of 102 to all OSPF routes received from 172.16.10.2.
`Router(config-router)#distance 95 172.16.20.0 0.0.0.255 2`	Assigns an AD of 95 to any routes matching ACL 2 that are learned from network 172.16.20.0.
`Router(config-router)#exit`	Returns to global configuration mode.
`Router(config)#access-list 2 permit 192.168.30.0 0.0.0.255`	Creates an ACL that will define what route or routes will have an AD of 95 assigned to it.
	NOTE: A named ACL can also be used. Replace the ACL number with the name of the ACL in this command: `Router(config-router)#distance 95 172.16.20.2 255.255.255.0 namedACL`

Static Routes: permanent Keyword

`Router(config)#ip route 192.168.50.0 255.255.255.0 serial0/0/0 permanent`	Creates a static route that will not be removed from the routing table, even if the interface shuts down for any reason.

Without the **permanent** keyword in a static route statement, a static route will be removed if the interface specified in the command goes down. A downed interface will cause the directly connected network and any associated static routes to be removed from the routing table. If the interface comes back up, the routes will be returned.

Adding the **permanent** keyword to a static route statement will keep the static routes in the routing table even if the interface goes down and the directly connected networks are removed. You *cannot* get to these routes—the interface is down—but the routes remain in the table. The advantage to this is that when the interface comes back up, the static routes do not need to be reprocessed and placed back into the routing table, saving time and processing power.

When a static route is added or deleted, this route, along with all other static routes, is processed in one second. Before Cisco IOS Release 12.0, this was 5 seconds.

The routing table processes static routes every minute to install or remove static routes according to the changing routing table.

Floating Static Routes

`Router(config)#ip route 192.168.50.0 255.255.255.0 serial0/0/0 130`	Creates a static route that has an AD of 130 rather than the default AD of 1
`Router(config)#ipv6 route 2001:db8:c18:3::/64 fastethernet0/0 200`	Creates an IPv6 static route that has an AD of 200 rather than the default AD of 1

TIP: By default, a static route will always be used rather than a routing protocol. By adding an AD number to your **ip route** statement, you can effectively create a backup route to your routing protocol. If your network is using EIGRP, and you need a backup route, add a static route with an AD greater than 90. EIGRP will be used because its AD is better (lower) than the static route. If EIGRP goes down, the static route is used in its place. When EIGRP is running again, EIGRP routes are used because their AD will again be lower than the AD of the floating static route.

Static Routes and Recursive Lookups

A static route that uses a next-hop address (intermediate address) will cause the router to look at the routing table twice: once when a packet first enters the router and the router looks up the entry in the table, and a second time when the router has to resolve the location of the intermediate address.

For point-to-point links, always use an exit interface in your static route statements:

```
Router(config)#ip route 192.168.10.0 255.255.255.0 serial0/0/0
```

For broadcast links such as Ethernet, Fast Ethernet, or Gigabit Ethernet, use *both* an exit interface and intermediate address:

```
Router(config)#ip route 192.168.10.0 255.255.255.0 fastethernet0/0
  192.138.20.2
```

This saves the router from having to do a recursive lookup for the intermediate address of 192.168.20.2, knowing that the exit interface is Fast Ethernet 0/0.

Try to avoid using static routes that reference only intermediate addresses.

Default Routes

NOTE: To create a default route in IPv6, you use the same format as creating a default route in IPv4.

`Router(config)#ip route 0.0.0.0 0.0.0.0 172.16.10.2 serial0/0/0`	Send all packets destined for networks not in the routing table to 172.16.10.2 out exit interface Serial 0/0/0
`Router(config)#ip route 0.0.0.0 0.0.0.0 serial0/0/0`	Send all packets destined for networks not in the routing table out the Serial 0/0/0 interface
`Austin(config)#ipv6 route ::/0 2001:db8:c18:2::2/6 serial0/0/0`	Creates a default route configured to send all packets not in the routing table to a next-hop address of 2001:db8:c18:2::2 out exit interface Serial 0/0/0
`Austin(config)#ipv6 route ::/0 gigabitethernet0/0`	Creates a default route configured to send all packets not in the routing table out interface GigabitEthernet 0/0

NOTE: The combination of the 0.0.0.0 network address and the 0.0.0.0 mask is called a *quad-zero route*.

Verifying Static Routes

To display the contents of the IP routing table, enter the following command:

```
Router#show ip route
```

or

```
Router#show ipv6 route
```

The codes to the left of the routes in the table tell you from where the router learned the routes. A static route is described by the letter *S*. A default route is described in the routing table by S*. The asterisk (*) indicates that this is a candidate default option that will be used when forwarding packets.

Assigning IPv6 Addresses to Interfaces

This section shows multiple ways to assign the various types of IPv6 addresses to an interface.

`Router(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router.
`Router(config)#interface gigabitethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#ipv6 enable`	Automatically configures an IPv6 link-local address on the interface and enables IPv6 processing on the interface.
	NOTE: The link-local address that the **ipv6 enable** command configures can be used only to communicate with nodes on the same link.
`Router(config-if)#ipv6 address autoconfig`	Router will configure itself with a link-local address using stateless auto configuration.
`Router(config-if)#ipv6 address 2001::1/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
	NOTE: If you add a global IPv6 address to the interface before entering the **ipv6 enable** command, a link-local address will automatically be created, and IPv6 will be enabled on the interface.
`Router(config-if)#ipv6 address 2001:db8:0:1::/64 eui-64`	Configures a global IPv6 address with an EUI-64 interface identifier in the low-order 64 bits of the IPv6 address.
`Router(config-if)#ipv6 address fe80::260:3eff:fe47:1530/64 link-local`	Configures a specific link-local IPv6 address on the interface instead of the one that is automatically configured when IPv6 is enabled on the interface.
`Router(config-if)#ipv6 unnumbered` *`type/number`*	Specifies an unnumbered interface and enables IPv6 processing on the interface. The global IPv6 address of the interface specified by *type/number* will be used as the source address for packets sent from the interface.

Implementing RIP Next Generation (RIPng)

This section shows how to implement RIPng on a router.

`Router(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router.
`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.

`Router(config-if)#ipv6 rip tower enable`	Creates the RIPng process named tower and enables RIPng on the interface.
	NOTE: Unlike RIPv1 and RIPv2, where you needed to create the RIP routing process with the **router rip** command and then use the **network** command to specify the interfaces on which to run RIP, the RIPng process is created automatically when RIPng is enabled on an interface with the **ipv6 rip** *name* **enable** command.
	TIP: Be sure that you do not misspell your process name. If you do misspell the name, you will inadvertently create a second process with the misspelled name.
	NOTE: Cisco IOS Software automatically creates an entry in the configuration for the RIPng routing process when it is enabled on an interface.
	NOTE: The **ipv6 router rip** *process-name* command is still needed when configuring optional features of RIPng.
	NOTE: The routing process name does not need to match between neighbor routers.
`Router(config)#ipv6 router rip tower`	Creates the RIPng process named *tower* if it has not already been created, and moves to router configuration mode.
`Router(config-router)#maximum-paths 2`	Defines the maximum number of equal-cost routes that RIPng can support.
	NOTE: The number of paths that can be used is a number from 1 to 64. The default is 4.
`Router(config-if)#ipv6 rip tower default-information originate`	Announces the default route along with all other RIPng routes.
`Router(config-if)#ipv6 rip tower default-information only`	Announces only the default route. Suppresses all other RIPng routes.

Verifying and Troubleshooting RIPng

CAUTION: Using the **debug** command may severely affect router performance and might even cause the router to reboot. Always exercise caution when using the **debug** command. Do not leave **debug** on. Use it long enough to gather needed information, and then disable debugging with the **undebug all** command.

TIP: Send your **debug** output to a syslog server to ensure you have a copy of it in case your router is overloaded and needs to reboot.

`Router#clear ipv6 rip`	Deletes routes from the IPv6 RIP routing table and, if installed, routes in the IPv6 routing table.
`Router#clear ipv6 route *`	Deletes all routes from the IPv6 routing table.
	NOTE: Clearing all routes from the routing table will cause high CPU utilization rates as the routing table is rebuilt.
`Router#clear ipv6 route 2001:db8:c18:3::/64`	Clears this specific route from the IPv6 routing table.
`Router#clear ipv6 traffic`	Resets IPv6 traffic counters.
`Router#debug ipv6 packet`	Displays debug messages for IPv6 packets.
`Router#debug ipv6 rip`	Displays debug messages for IPv6 RIP routing transactions.
`Router#debug ipv6 routing`	Displays debug messages for IPv6 routing table updates and route cache updates.
`Router#show ipv6 interface`	Displays the status of interfaces configured for IPv6.
`Router#show ipv6 interface brief`	Displays a summarized status of interfaces configured for IPv6.
`Router#show ipv6 neighbors`	Displays IPv6 neighbor discovery cache information.
`Router#show ipv6 protocols`	Displays the parameters and current state of the active IPv6 routing protocol processes.
`Router#show ipv6 rip`	Displays information about the current IPv6 RIPng process.
`Router#show ipv6 rip database`	Displays the RIPng process database. If more than one RIPng process is running, all will be displayed with this command.
`Router#show ipv6 rip next-hops`	Displays RIPng processes and, under each process, all next-hop addresses.
`Router#show ipv6 route`	Displays the current IPv6 routing table.
`Router#show ipv6 route rip`	Displays the current RIPng routes in the IPv6 routing table
`Router#show ipv6 route summary`	Displays a summarized form of the current IPv6 routing table.
`Router#show ipv6 routers`	Displays IPv6 router advertisement information received from other routers.
`Router#show ipv6 traffic`	Displays statistics about IPv6 traffic.

Configuration Example: RIPng

Figure 1-5 illustrates the network topology for the configuration that follows, which shows how to configure IPv6 and RIPng using the commands covered in this chapter.

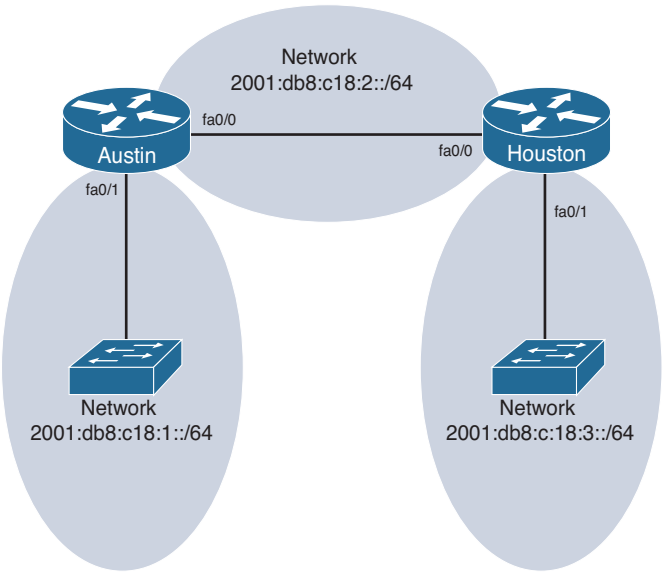


Figure 1-5 Network Topology for IPv6/RIPng Configuration Example

Austin Router

`Router>enable`	Moves to privileged mode
`Router#configure terminal`	Moves to global configuration mode
`Router(config)#hostname Austin`	Assigns a host name to the router
`Austin(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router
`Austin(config)#interface fastethernet0/0`	Enters interface configuration mode
`Austin(config-if)#ipv6 address 2001:db8:c18:2::/64 eui-64`	Configures a global IPv6 address with an EUI-64 interface identifier in the low-order 64 bits of the IPv6 address
`Austin(config-if)#ipv6 rip tower enable`	Creates the RIPng process named tower and enables RIPng on the interface
`Austin(config-if)#no shutdown`	Activates the interface
`Austin(config-if)#interface fastethernet0/1`	Enters interface configuration mode
`Austin(config-if)#ipv6 address 2001:db8:c18:1::/64 eui-64`	Configures a global IPv6 address with an EUI-64 interface identifier in the low-order 64 bits of the IPv6 address
`Austin(config-if)#ipv6 rip tower enable`	Creates the RIPng process named tower and enables RIPng on the interface
`Austin(config-if)#no shutdown`	Activates the interface

`Austin(config-if)#exit`	Moves to global configuration mode
`Austin(config)#exit`	Moves to privileged mode
`Austin#copy running-config startup-config`	Saves the configuration to NVRAM

Houston Router

`Router>enable`	Moves to privileged mode
`Router#configure terminal`	Moves to global configuration mode
`Router(config)#hostname Houston`	Assigns a host name to the router
`Houston(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router
`Houston(config)#interface fastethernet0/0`	Enters interface configuration mode
`Houston(config-if)#ipv6 address 2001:db8:c18:2::/64 eui-64`	Configures a global IPv6 address with an EUI-64 interface identifier in the low-order 64 bits of the IPv6 address
`Houston(config-if)#ipv6 rip tower enable`	Creates the RIPng process named tower and enables RIPng on the interface
`Houston(config-if)#no shutdown`	Activates the interface
`Houston(config-if)#interface fastethernet 0/1`	Enters interface configuration mode
`Houston(config-if)#ipv6 address 2001:db8:c18:3::/64 eui-64`	Configures a global IPv6 address with an EUI-64 interface identifier in the low-order 64 bits of the IPv6 address
`Houston(config-if)#ipv6 rip tower enable`	Creates the RIPng process named tower and enables RIPng on the interface
`Houston(config-if)#no shutdown`	Activates the interface
`Houston(config-if)#exit`	Moves to global configuration mode
`Houston(config)#exit`	Moves to privileged mode
`Houston#copy running-config startup-config`	Saves the configuration to NVRAM

IPv6 Ping

To diagnose basic network connectivity using IPv6 to the specified address, enter the **ping** command as shown in the following example:

```
Router#ping ipv6 2001:db8::3/64
```

The following characters can be displayed as output when using ping in IPv6.

Character	Description
!	Each exclamation point indicates receipt of a reply.
.	Each period indicates that the network server timed out while waiting for a reply.
?	Unknown error.
@	Unreachable for unknown reason.
A	Administratively unreachable. Usually means that an access control list (ACL) is blocking traffic.
B	Packet too big.
H	Host unreachable.
N	Network unreachable (beyond scope).
P	Port unreachable.
R	Parameter problem.
T	Time exceeded.
U	No route to host.

IPv6 Traceroute

To observe the path between two hosts using IPv6 to the specified address, the **traceroute** command in Cisco IOS or the **tracert** Windows command may be used, as shown in the following examples:

```
Router#traceroute 2001:db8:c18:2::1
C:\Windows\system32>tracert 2001:DB8:c18:2::1
```

CHAPTER 2

EIGRP Implementation

This chapter provides information about the following topics:

- Configuring EIGRP
- EIGRP router ID
- EIGRP autosummarization
- Passive EIGRP interfaces
- "Pseudo" passive EIGRP interfaces
- EIGRP timers
- Injecting a default route into EIGRP: redistribution of a static route
- Injecting a default route into EIGRP: IP default network
- Injecting a default route into EIGRP: summarize to 0.0.0.0/0
- Accepting exterior routing information: default-information
- Load balancing: maximum paths
- Load balancing: variance
- Bandwidth use
- Stub networks
- EIGRP unicast neighbors
- EIGRP over Frame Relay: dynamic mappings
- EIGRP over Frame Relay: static mappings
- EIGRP over Frame Relay: EIGRP over multipoint subinterfaces
- EIGRP over Frame Relay: EIGRP over point-to-point subinterfaces
- EIGRP over MPLS: Layer 2 VPN
- EIGRP over MPLS: Layer 3 VPN
- EIGRPv6
    - Enabling EIGRPv6 on an interface
    - Configuring the percentage of link bandwidth used by EIGRPv6
    - EIGRPv6 summary addresses
    - EIGRPv6 timers
    - EIGRPv6 stub routing
    - Logging EIGRPv6 neighbor adjacency changes
    - Adjusting the EIGRPv6 metric weights
- EIGRP address families
- Verifying EIGRP
- Troubleshooting EIGRP
- Configuration example: EIGRPv4 and EIGRPv6 using named address configurations

Configuring EIGRP

`Router(config)#router eigrp 100`	Turns on the EIGRP process. 100 is the autonomous system number, which can be a number between 1 and 65,535. All routers in the same autonomous system must use the same autonomous system number.
`Router(config-router)#network 10.0.0.0`	Specifies which network to advertise in EIGRP.
`Router(config-router)#network 10.0.0.0 0.255.255.255`	Identifies which interfaces or networks to include in EIGRP. Interfaces must be configured with addresses that fall within the wildcard mask range of the **network** statement.
	NOTE: The use of a wildcard mask is optional.
	NOTE: There is no limit to the number of **network** statements (that is, **network** commands) that you can configure on a router.

TIP: If you are using the **network 172.16.1.0 0.0.0.255** command with a wildcard mask, in this example the command specifies that only interfaces on the 172.16.1.0/24 subnet will participate in EIGRP.

NOTE: If you do not use the optional wildcard mask, the EIGRP process assumes that all directly connected networks that are part of the overall major network will participate in the EIGRP process and EIGRP will attempt to establish neighbor relationships from each interface that is part of that Class A, B, or C major network.

`Router(config-router)#eigrp log-neighbor-changes`	Displays changes with neighbors.
`Router(config-router)#eigrp log-neighbor-warnings 300`	Configures the logging intervals of EIGRP neighbor warning messages to 300 seconds. The default is 10 seconds.
`Router(config-if)#bandwidth 256`	Sets the bandwidth of this interface to 256 kilobits to allow EIGRP to make a better metric calculation.
	TIP: The **bandwidth** command is used for metric calculations only. It does not change interface performance.
`Router(config-router)#no network 10.0.0.0`	Removes the network from the EIGRP process.
	NOTE: If you used the optional wildcard mask in the original command it needs to be added here as well.
`Router(config)#no router eigrp 100`	Disables routing process 100.

`Router(config-router)#`**`metric weights`** *`tos k1 k2 k3 k4 k5`* `Router(config-router)#`**`metric weights 0 1 1 1 1 1`**	Changes the default k values used in metric calculation. These are the default values: tos=0, k1=1, k2=0, k3=1, k4=0, k5=0.

NOTE: *tos* is a reference to the original Interior Gateway Routing Protocol (IGRP) intention to have IGRP perform type of service routing. Because this was never adopted into practice, the tos field in this command is always set to 0.

NOTE: With default settings in place, the metric of Enhanced Interior Gateway Routing Protocol (EIGRP) is reduced to using the slowest bandwidth along the path, plus the sum of all the delays of the exit interfaces from the local router to the destination network.

TIP: For two routers to form a neighbor relationship in EIGRP, the k values must match.

NOTE: Unless you are very familiar with what is occurring in your network, it is recommended that you do not change the k values.

EIGRP Router ID

`Router(config)#`**`router eigrp 100`**	Enters into EIGRP router configuration mode for autonomous system 100.
`Router(config-router)#` **`eigrp router-id 172.16.3.3`**	Manually sets the router ID to 172.16.3.3. Can be any IP address except for 0.0.0.0 and 255.255.255.255. If not set, the router ID will be the highest IP address of any loopback interfaces. If no loopback interfaces are configured, the router ID will be the highest IP address of your active local interface.
`Router(config-router)#`**`no eigrp router-id 172.16.3.3`**	Removes the static router ID from the configuration.

EIGRP Autosummarization

`Router(config-router)#`**`auto-summary`**	Enables autosummarization for the EIGRP process.
`Router(config-router)#`**`no auto-summary`**	Turns off the autosummarization feature.
`Router(config)#`**`interface fastethernet 0/0`**	Enters interface configuration mode.

`Router(config-if)#ip summary-address eigrp 100 10.10.0.0 255.255.0.0 75`	Enables manual summarization for EIGRP autonomous system 100 on this specific interface for the given address and mask. An administrative distance (AD) of 75 is assigned to this summary route.
	NOTE: The **administrative-distance** argument is optional in this command. Without it, an administrative distance of 5 is automatically applied to the summary route.
	NOTE: The AD of 5 will only be shown with the **show ip route 10.10.0.0 255.255.0.0** command.

NOTE: EIGRP no longer automatically summarizes networks at the classful boundary by default, since Cisco IOS Software Release 15.0.

CAUTION: Recommended practice is that you turn off automatic summarization if necessary, use the **ip summary-address** command, and summarize manually what you need to. A summary route will have the metric of the subnet with the lowest metric.

Passive EIGRP Interfaces

`Router(config)#router eigrp 100`	Starts the EIGRP routing process.
`Router(config-router)#network 10.0.0.0`	Specifies a network to advertise in the EIGRP routing process.
`Router(config-router)#passive-interface fastethernet0/0`	Prevents the sending of hello packets out the Fast Ethernet 0/0 interface. No neighbor adjacency will be formed.
	NOTE: The router will still advertise the subnet for the passive interface.
	TIP: Passive interfaces are useful when you have interfaces connected to end devices.
`Router(config-router)#passive-interface default`	Prevents the sending of hello packets out all interfaces.
`Router(config)#no passive-interface serial0/0/1`	Enables hello packets to be sent out interface Serial 0/0/1, thereby allowing neighbor adjacencies to form.

"Pseudo" Passive EIGRP Interfaces

NOTE: A passive interface cannot send EIGRP hellos, which prevents adjacency relationships with link partners.

NOTE: An administrator can create a "pseudo" passive EIGRP interface by using a route filter that suppresses all routes from the EIGRP routing update. A neighbor relationship will form, but no routes will be sent out a specific interface.

`Router(config)#router eigrp 100`	Starts the EIGRP routing process.
`Router(config-router)#network 10.0.0.0`	Specifies a network to advertise in the EIGRP routing process.
`Router(config-router)#distribute-list 5 out serial0/0/0`	Creates an outgoing distribute list for interface Serial 0/0/0 and refers to ACL 5.
`Router(config-router)#exit`	Returns to global configuration mode.
`Router(config)#access-list 5 deny any`	This ACL, when used in the earlier **distribute-list** command, will cause no EIGRP 100 routing packets to be sent out s0/0/0.

EIGRP Timers

`Router(config-if)#ip hello-interval eigrp 100 10`	Configures the EIGRP hello time interval for autonomous system 100 to 10 seconds
`Router(config-if)#ip hold-time eigrp 100 30`	Configures the EIGRP hold timer interval for autonomous system 100 to 30 seconds

NOTE: EIGRP hello and hold timers do not have to match between neighbors to successfully establish a neighbor relationship.

NOTE: The autonomous system number in these commands must match the autonomous system number of EIGRP on the router for these changes to take effect.

TIP: It is recommended that you match the timers between neighbors or you may experience flapping neighbor relationships/network instability.

Injecting a Default Route into EIGRP: Redistribution of a Static Route

`Router(config)#ip route 0.0.0.0 0.0.0.0 serial0/0/0`	Creates a static default route to send all traffic with a destination network not in the routing table out interface Serial 0/0/0.
	NOTE: Adding a static route (for example, **ip route 0.0.0.0 0.0.0.0 fastethernet1/2**) will cause the route to be inserted into the routing table only when the interface is up.
`Router(config)#router eigrp 100`	Creates EIGRP routing process 100.
`Router(config-router)#redistribute static`	Static routes on this router will be exchanged with neighbor routers in EIGRP.

NOTE: Use this method when you want to draw all traffic to unknown destinations to a default route at the core of the network.

NOTE: This method is effective for advertising default connections to the Internet, but it will also redistribute all static routes into EIGRP.

Injecting a Default Route into EIGRP: IP Default Network

`Router(config)#router eigrp 100`	Creates EIGRP routing process 100
`Router(config-router)#network 192.168.100.0`	Specifies which network to advertise in EIGRP
`Router(config-router)#exit`	Returns to global configuration mode
`Router(config)#ip route 0.0.0.0 0.0.0.0 192.168.100.5`	Creates a static default route to send all traffic with a destination network not in the routing table to next-hop address 192.168.100.5
`Router(config)#ip default-network 192.168.100.0`	Defines a route to the 192.168.100.0 network as a candidate default route

NOTE: For EIGRP to propagate the route, the network specified by the **ip default-network** command must be known to EIGRP. This means that the network must be an EIGRP-derived network in the routing table, or the static route used to generate the route to the network must be redistributed into EIGRP, or advertised into these protocols using the **network** command.

TIP: In a complex topology, many networks can be identified as candidate defaults. Without any dynamic protocols running, you can configure your router to choose from a number of candidate default routes based on whether the routing table has routes to networks other than 0.0.0.0/0. The **ip default-network** command enables you to configure robustness into the selection of a gateway of last resort. Rather than configuring static routes to specific next hops, you can have the router choose a default route to a particular network by checking in the routing table.

TIP: You can propagate the 0.0.0.0 network through EIGRP by using the **network 0.0.0.0** statement.

TIP: The **network 0.0.0.0** command enables EIGRP for all interfaces on the router.

Injecting a Default Route into EIGRP: Summarize to 0.0.0.0/0

`Router(config)#router eigrp 100`	Creates EIGRP routing process 100.
`Router(config-router)#network 192.168.100.0`	Specifies which network to advertise in EIGRP.
`Router(config-router)#exit`	Returns to global configuration mode.
`Router(config)#interface serial0/0/0`	Enters interface configuration mode.
`Router(config-if)#ip address 192.168.100.1 255.255.255.0`	Assigns the IP address and subnet mask to the interface.
`Router(config-if)#ip summary-address eigrp 100 0.0.0.0 0.0.0.0 75`	Enables manual summarization for EIGRP autonomous system 100 on this specific interface for the given address and mask. An optional administrative distance of 75 is assigned to this summary route.

NOTE: Summarizing to a default route is effective only when you want to provide remote sites with a default route, and not propagate the default route toward the core of your network.

NOTE: Because summaries are configured per interface; you do not need to worry about using distribute lists or other mechanisms to prevent the default route from being propagated toward the core of your network.

Accepting Exterior Routing Information: default-information

`Router(config)#router eigrp 100`	Creates routing process 100.
`Router(config-router)#default-information in`	Allows exterior or default routes to be received by the EIGRP process autonomous system 100. This is the default action; exterior routes are always accepted and default information is passed between EIGRP processes when redistribution occurs.
`Router(config-router)#no default-information in`	Suppresses exterior or default routing information.

Load Balancing: Maximum Paths

`Router(config)#router eigrp 100`	Creates routing process 100.
`Router(config-router)#network 10.0.0.0`	Specifies which network to advertise in EIGRP.
`Router(config-router)#maximum-paths 3`	Sets the maximum number of equal metric routes that EIGRP will support to three. If the **variance** command is used (as described in the following section), unequal metric paths will also be included.

NOTE: The maximum number of paths and default number of paths varies by IOS.

NOTE: Setting the **maximum-path** to **1** disables load balancing.

Load Balancing: Variance

`Router(config)#router eigrp 100`	Creates routing process 100.
`Router(config-router)#network 10.0.0.0`	Specifies which network to advertise in EIGRP.
`Router(config-router)#variance 3`	Instructs the router to include routes with a metric less than 3 times the minimum metric route for that destination. The variance parameter can be a number between 1 and 128.

NOTE: If a path is not a feasible successor, it is not used in load balancing.

NOTE: To control how traffic is distributed among routes when there are multiple routes for the same destination network that have different costs, use the **traffic-share balanced** command. Traffic is distributed proportionately to the ratio of the costs by default.

Bandwidth Use

`Router(config)#interface serial0/0/0`	Enters interface configuration mode.
`Router(config-if)#bandwidth 256`	Sets the bandwidth of this interface to 256 kilobits; this command sets the bandwidth used in the EIGRP metric calculation.
`Router(config-if)#ip bandwidth-percent eigrp 100 75`	Configures the percentage of bandwidth that may be used by EIGRP on an interface. 100 is the EIGRP autonomous system number. 75 is the percentage value. 75% * 256 = 192 Kbps.

NOTE: By default, EIGRP is set to use only up to 50 percent of the bandwidth of an interface to exchange routing information. Values greater than 100 percent can be configured. This configuration option might prove useful if the bandwidth is set artificially low for other reasons, such as manipulation of the routing metric or to accommodate an oversubscribed multipoint Frame Relay configuration.

NOTE: The **ip bandwidth-percent** command relies on the value set by the **bandwidth** command.

Stub Networks

`Router(config)#router eigrp 100`	Creates routing process 100.
`Router(config-router)#eigrp stub`	Prompts the router to send updates containing its connected and summary routes only.
	NOTE: Only the stub router needs to have the **eigrp stub** command enabled.
`Router(config-router)#eigrp stub connected`	Permits the EIGRP stub routing feature to send only connected routes.
	NOTE: If the connected routes are not covered by a **network** statement, it might be necessary to redistribute connected routes with the **redistribute connected** command.

	TIP: The **connected** option is enabled by default.
`Router(config-router)#`**`eigrp stub static`**	Permits the EIGRP stub routing feature to send static routes.
	NOTE: Without this option, EIGRP will not send static routes, including internal static routes that normally would be automatically redistributed. It will still be necessary to redistribute static routes with the **redistribute static** command.
`Router(config-router)#`**`eigrp stub summary`**	Permits the EIGRP stub routing feature to send summary routes.
	NOTE: Summary routes can be created manually, or through automatic summarization at a major network boundary if the **auto-summary** command is enabled.
	TIP: The **summary** option is enabled by default.
`Router(config-router)#`**`eigrp stub receive-only`**	Restricts the router from sharing any of its routes with any other router in that EIGRP autonomous system.
`Router(config-router)#`**`eigrp stub redistributed`**	Advertises redistributed routes, if redistribution is configured on the stub router using the **redistribute** command

NOTE: You can use the optional arguments (**connected**, **redistributed**, **static**, and **summary**) as part of the same command on a single line:

```
Router(config-router)#eigrp stub connected static summary
  redistributed
```

You cannot use the keyword **receive-only** with any other option because it prevents any type of route from being sent.

EIGRP Unicast Neighbors

`R2(config)#`**`router eigrp 100`**	Enables EIGRP routing for autonomous system 100.
`R2(config-router)#` **`network 192.168.1.0`**	Identifies which networks to include in EIGRP.
`R2(config-router)#` **`neighbor 192.168.1.101 fastethernet0/0`**	Identifies a specific neighbor with which to exchange routing information. Instead of using multicast packets to exchange information, unicast packets will now be used on the interface on which this neighbor resides. If there are other neighbors on this same interface, **neighbor** statements must also be configured for them; otherwise, no EIGRP packets will be exchanged with them.

EIGRP over Frame Relay: Dynamic Mappings

Figure 2-1 shows the network topology for the configuration that follows, which shows how to configure EIGRP over Frame Relay using dynamic mappings.

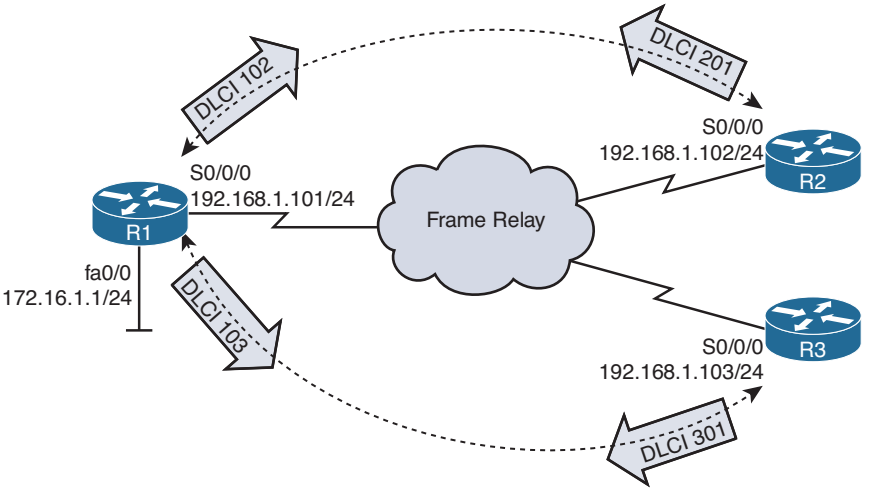


Figure 2-1 Network Topology for EIGRP over Frame Relay Using Dynamic Mappings

`R1(config)#interface serial0/0/0`	Enters interface configuration mode
`R1(config-if)#ip address 192.168.1.101 255.255.255.0`	Assigns the IP address and mask
`R1(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface
`R1(config-if)#no shutdown`	Enables the interface
`R1(config-if)#exit`	Returns to global configuration mode
`R1(config)#router eigrp 100`	Creates routing process 100
`R1(config-router)#network 172.16.1.0 0.0.0.255`	Advertises the network in EIGRP
`R1(config-router)#network 192.168.1.0`	Advertises the network in EIGRP

NOTE: To deploy EIGRP over a physical interface using dynamic mappings—relying on Inverse ARP—no changes are needed to the basic EIGRP configuration.

NOTE: In EIGRP, split horizon is disabled by default on Frame Relay physical interfaces. Therefore, R2 and R3 can provide connectivity between their connected networks. Inverse ARP does not provide dynamic mappings for communication between R2 and R3; this must be configured manually.

EIGRP over Frame Relay: Static Mappings

Figure 2-2 shows the network topology for the configuration that follows, which shows how to configure EIGRP over Frame Relay using static mappings.

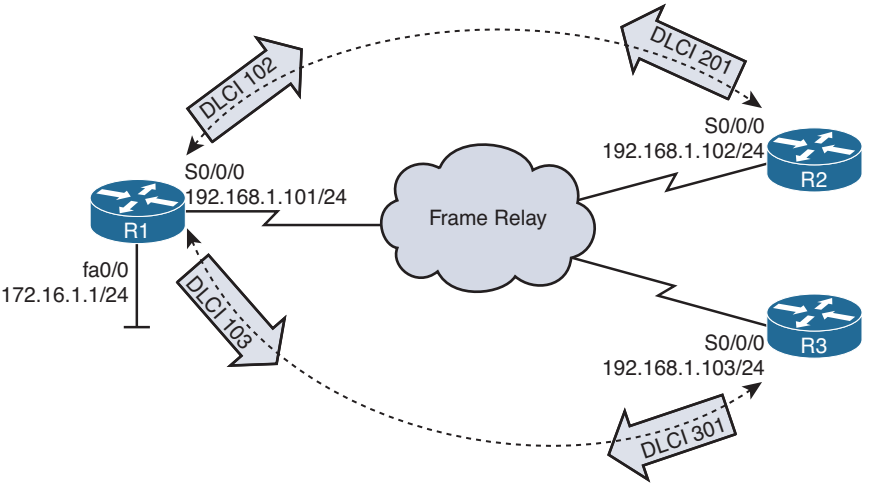


Figure 2-2 Network Topology for EIGRP over Frame Relay Using Static Mappings

`R1(config)#interface serial0/0/0`	Enters interface configuration mode.
`R1(config-if)#ip address 192.168.1.101 255.255.255.0`	Assigns the IP address and mask.
`R1(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface.
`R1(config-if)#frame-relay map ip 192.168.1.101 102`	Maps the IP address of 192.168.1.101 to DLCI 102.
	NOTE: The router includes this map to its own IP address so that the router can ping the local address from itself.
`R1(config-if)#frame-relay map ip 192.168.1.102 102 broadcast`	Maps the remote IP address 192.168.1.102 to DLCI 102. The **broadcast** keyword means that broadcasts and multicasts will now be forwarded as well.
`R1(config-if)#frame-relay map ip 192.168.1.103 103 broadcast`	Maps the remote IP address 192.168.1.103 to DLCI 103. The **broadcast** keyword means that broadcasts and multicasts will now be forwarded as well.
`R1(config-if)#no shutdown`	Enables the interface.
`R1(config-if)#exit`	Returns to global configuration mode.
`R1(config)#router eigrp 100`	Creates routing process 100.

`R1(config-router)#network 172.16.1.0 0.0.0.255`	Advertises the network in EIGRP.
`R1(config-router)#network 192.168.1.0`	Advertises the network in EIGRP.

NOTE: To deploy EIGRP over a physical interface using static mappings—and thus disabling Inverse ARP—no changes are needed to the basic EIGRP configuration. Only manual IP to data link connection identifier (DLCI) mapping statements are required on all three routers.

NOTE: In EIGRP, split horizon is disabled by default on Frame Relay physical interfaces. Therefore, R2 and R3 can provide connectivity between their connected networks. Inverse ARP does not provide dynamic mappings for communication between R2 and R3; this must be configured manually.

EIGRP over Frame Relay: EIGRP over Multipoint Subinterfaces

Figure 2-3 shows the network topology for the configuration that follows, which shows how to configure EIGRP over Frame Relay using multipoint subinterfaces.

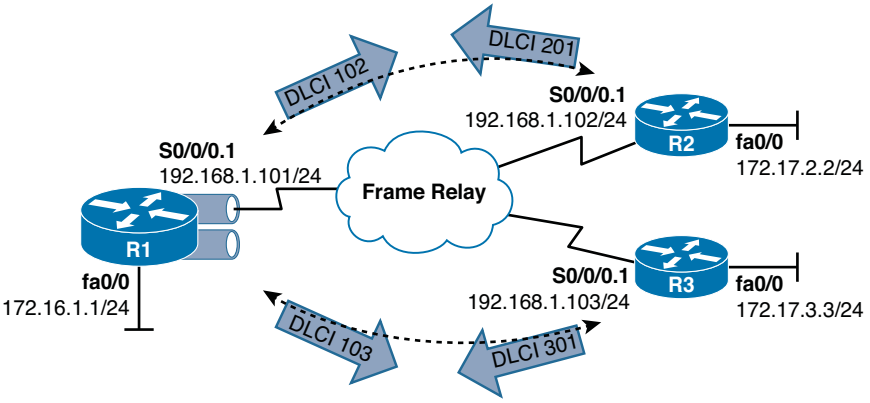


Figure 2-3 Network Topology for EIGRP over Frame Relay Using Multipoint Subinterfaces

`R1(config)#interface serial0/0/0`	Enters interface configuration mode.
`R1(config-if)#no ip address`	Removes any previous IP address and mask information assigned to this interface. Interface now has no address or mask.
`R1(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface.

`R1(config-if)#no frame-relay inverse-arp eigrp 100`	Turns off dynamic mapping for EIGRP 100.
`R1(config-if)#exit`	Returns to global configuration mode.
`R1(config)#interface serial0/0/0.1 multipoint`	Enables subinterface configuration mode. Multipoint behavior is also enabled.
`R1(config-subif)#ip address 192.168.1.101 255.255.255.0`	Assigns IP address and mask information.
`R1(config-subif)#no ip split-horizon eigrp 100`	Disables split horizon for EIGRP on this interface. This is to allow R2 and R3 to have connectivity between their connected networks.
`R1(config-subif)#frame-relay map ip 192.168.1.101 102`	Maps the IP address of 192.168.1.101 to DLCI 102.
	NOTE: The router includes this map to its own IP address so that the router can ping the local address from itself.
`R1(config-subif)#frame-relay map ip 192.168.1.102 102 broadcast`	Maps the remote IP address 192.168.1.102 to DLCI 102. The **broadcast** keyword means that broadcasts and multicasts will now be forwarded as well.
`R1(config-subif)#frame-relay map ip 192.168.1.103 103 broadcast`	Maps the remote IP address 192.168.1.103 to DLCI 103. The **broadcast** keyword means that broadcasts and multicasts will now be forwarded as well.
`R1(config-subif)#exit`	Returns to global configuration mode.
`R1(config)#router eigrp 100`	Creates routing process 100.
`R1(config-router)#network 172.16.1.0 0.0.0.255`	Advertises the network in EIGRP.
`R1(config-router)#network 192.168.1.0`	Advertises the network in EIGRP.

NOTE: To deploy EIGRP over multipoint subinterfaces, no changes are needed to the basic EIGRP configuration.

EIGRP over Frame Relay: EIGRP over Point-to-Point Subinterfaces

Figure 2-4 shows the network topology for the configuration that follows, which shows how to configure EIGRP over Frame Relay using point-to-point subinterfaces.

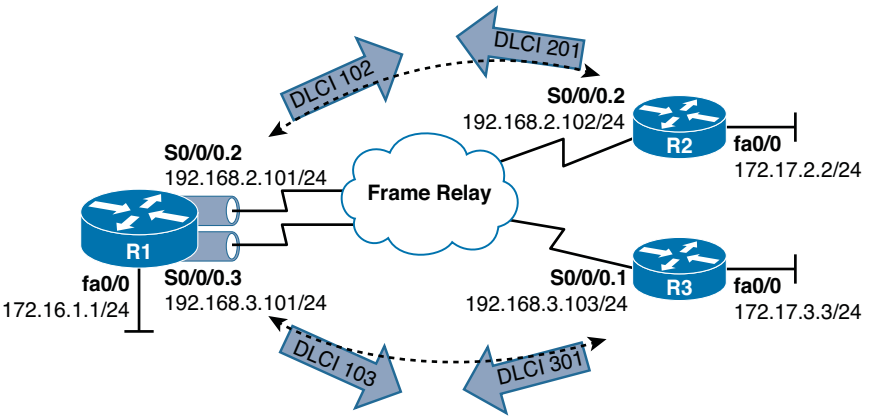


Figure 2-4 Network Topology for EIGRP over Frame Relay Using Point-to-Point Subinterfaces

R1 Router

`R1(config)#interface serial0/0/0`	Enters interface configuration mode.
`R1(config-if)#no ip address`	Removes any previous IP address and mask information assigned to this interface. Interface now has no address or mask.
`R1(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface.
`R1(config-if)#exit`	Returns to global configuration mode.
`R1(config)#interface serial0/0/0.2 point-to-point`	Enables subinterface configuration mode. Point-to-point behavior is also enabled.
`R1(config-subif)#ip address 192.168.2.101 255.255.255.0`	Assigns an IP address and mask to the subinterface.
`R1(config-subif)#frame-relay interface-dlci 102`	Assigns a local DLCI to this interface.
`R1(config-subif)#exit`	Returns to global configuration mode.
`R1(config)#interface serial0/0/0.3 point-to-point`	Enables subinterface configuration mode. Also enables point-to-point behavior.
`R1(config-subif)#ip address 192.168.3.101 255.255.255.0`	Assigns an IP address and mask to the subinterface.
`R1(config-subif)#frame-relay interface-dlci 103`	Assigns a local DLCI to this interface.
`R1(config-subif)#exit`	Returns to global configuration mode.
`R1(config)#router eigrp 100`	Creates routing process 100.
`R1(config-router)#network 172.16.1.0 0.0.0.255`	Advertises the network in EIGRP.

`R1(config-router)#network 192.168.2.0`	Advertises the network in EIGRP.
`R1(config-router)#network 192.168.3.0`	Advertises the network in EIGRP.

R3 Router

`R3(config)#interface serial0/0/0`	Enters interface configuration mode.
`R3(config-if)#no ip address`	Removes any previous IP address and mask information assigned to this interface. Address now has no address or mask.
`R3(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface.
`R3(config-if)#exit`	Returns to global configuration mode.
`R3(config)#interface serial0/0/0.1 point-to-point`	Enables subinterface configuration mode. Also enables point-to-point behavior.
`R3(config-subif)#ip address 192.168.3.103 255.255.255.0`	Assigns an IP address and mask to the subinterface.
`R3(config-subif)#frame-relay interface-dlci 103`	Assigns a local DLCI to this interface.
`R3(config-subif)#exit`	Returns to global configuration mode.
`R3(config)#router eigrp 100`	Creates routing process 100.
`R3(config-router)#network 172.16.3.0 0.0.0.255`	Advertises the network in EIGRP.
`R3(config-router)#network 192.168.3.0`	Advertises the network in EIGRP.

NOTE: To deploy EIGRP over point-to-point subinterfaces, no changes are needed to the basic EIGRP configuration.

EIGRP over MPLS: Layer 2 VPN

Figure 2-5 shows the network topology for the configuration that follows, which shows how to configure EIGRP over MPLS.

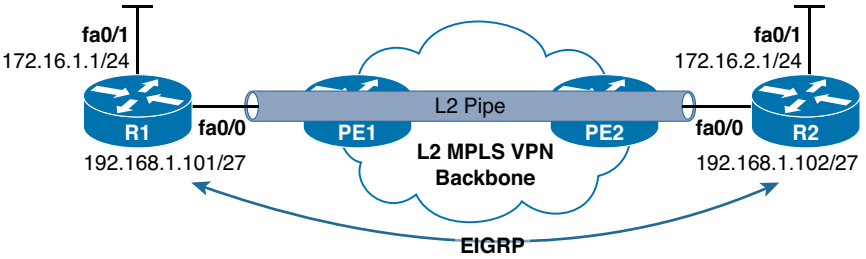


Figure 2-5 Network Topology for EIGRP over MPLS

NOTE: In this example, it is assumed that the MPLS network is configured with transparent Layer 2 transport, and only the EIGRP configuration is shown here.

R1 Router

`R1(config)#interface fastethernet0/0`	Enters interface configuration mode
`R1(config-if)#ip address 192.168.1.101 255.255.255.224`	Assigns the IP address and mask
`R1(config-if)#no shutdown`	Enables the interface
`R1(config-if)#router eigrp 100`	Creates routing process 100
`R1(config-router)#network 172.16.1.0 0.0.0.255`	Advertises the network in EIGRP
`R1(config-router)#network 192.168.1.0 0.0.0.255`	Advertises the network in EIGRP

R2 Router

`R2(config)#interface fastethernet0/0`	Enters interface configuration mode
`R2(config-if)#ip address 192.168.1.102 255.255.255.224`	Assigns the IP address and mask
`R2(config-if)#no shutdown`	Enables the interface
`R2(config-if)#router eigrp 100`	Creates routing process 100
`R2(config-router)#network 172.17.2.0 0.0.0.255`	Advertises the network in EIGRP
`R2(config-router)#network 192.168.1.0 0.0.0.255`	Advertises the network in EIGRP

NOTE: When deploying EIGRP over Multiprotocol Label Switching (MPLS), no changes are needed to the basic EIGRP configuration from the customer perspective.

NOTE: From the EIGRP perspective, the MPLS backbone and routers PE1 and PE2 are not visible. A neighbor relationship is established directly between routers R1 and R2; you can verify this with the **show ip eigrp neighbors** command output.

EIGRP over MPLS: Layer 3 VPN

Figure 2-6 shows the network topology for the configuration that follows, which shows how to configure EIGRP over MPLS where the MPLS PE devices are taking part in the EIGRP process.

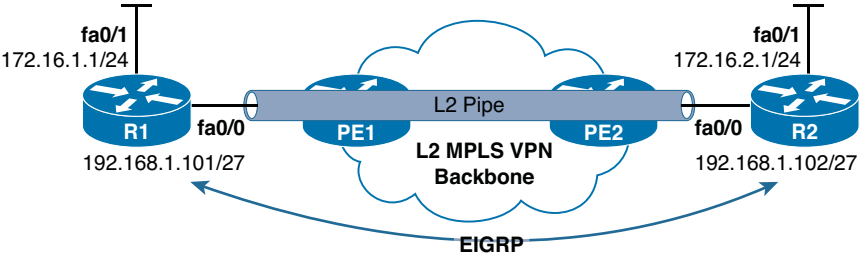


Figure 2-6 Network Topology for EIGRP over MPLS Layer 3 VPN

NOTE: In this example, it is assumed that the MPLS network is configured with the MPLS PE devices participating in the EIGRP process and virtual route forwarding. Only the client-side EIGRP configuration is shown here.

R1 Router

`R1(config)#interface fastethernet0/0`	Enters interface configuration mode
`R1(config-if)#ip address 192.168.1.2 255.255.255.252`	Assigns the IP address and mask
`R1(config-if)#no shutdown`	Enables the interface
`R1(config-if)#router eigrp 100`	Creates routing process 100
`R1(config-router)#network 172.16.1.0 0.0.0.255`	Advertises the network in EIGRP
`R1(config-router)#network 192.168.1.0 0.0.0.255`	Advertises the network in EIGRP

R2 Router

`R2(config)#interface fastethernet0/0`	Enters interface configuration mode
`R2(config-if)#ip address 192.168.2.2 255.255.255.252`	Assigns the IP address and mask
`R2(config-if)#no shutdown`	Enables the interface
`R2(config-if)#router eigrp 100`	Creates routing process 100
`R2(config-router)#network 172.17.2.0 0.0.0.255`	Advertises the network in EIGRP
`R2(config-router)#network 192.168.2.0 0.0.0.255`	Advertises the network in EIGRP

NOTE: When deploying EIGRP over Layer 3 MPLS, no changes are needed to the basic EIGRP configuration from the customer perspective. The only difference here is that the customer has to agree on the EIGRP parameters—autonomous system numbers, authentication password, and so on—with the service provider, because these parameters are often governed by the service provider.

NOTE: The PE routers receive IPv4 routing updates from the Client routers and install them in the appropriate Virtual Routing and Forwarding (VRF) table. This part of the configuration and operation is the responsibility of the service provider.

NOTE: From the EIGRP perspective, the MPLS backbone and routers PE1 and PE2 are not visible. A neighbor relationship is established directly between routers R1 and R2; you can verify this with the **show ip eigrp neighbors** command output.

EIGRPv6

No linkage exists between EIGRP for IPv4 and EIGRP for IPv6; they are configured and managed separately. However, the commands for configuration of EIGRP for IPv4 and IPv6 are very similar, making the transition very easy.

Enabling EIGRPv6 on an Interface

`Router(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router. This command is required before any IPv6 routing protocol can be configured.
`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.
`Router(config-if)#ipv6 eigrp 100`	Enables EIGRP for IPv6 on the interface, and creates the EIGRP for IPv6 process.
`Router(config-if)#ipv6 router eigrp 100`	Enters router configuration mode and creates an EIGRP IPv6 routing process.
`Router(config-router)#eigrp router-id 10.1.1.1`	Enables the use of a fixed router ID.

NOTE: Use the **eigrp router-id w.x.y.z** command only if an IPv4 address is not defined on the router eligible for router ID.

NOTE: EIGRP for IPv6 can also be created by entering into router configuration mode and creating the router process, just like you would with EIGRP for IPv4.

```
Router(config)#ipv6 router eigrp 400
Router(config-router)#eigrp router-id 10.1.1.1
```

Configuring the Percentage of Link Bandwidth Used by EIGRPv6

`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.
`Router(config-if)#ipv6 bandwidth-percent eigrp 100 75`	Configures the percentage of bandwidth (75%) that may be used by EIGRP for IPv6 on the interface. 100 is the EIGRP autonomous system number. 75 is the percentage value. This command behaves the same way as the **ip bandwidth-percentage eigrp** command.

EIGRPv6 Summary Addresses

`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.
`Router(config-if)#ipv6 summary-address eigrp 100 2001:0DB8:0:1::/64`	Configures a summary aggregate address for a specified interface. There is an optional administrative distance parameter for this command. This command behaves similar to the **ip summary-address eigrp** command.

EIGRPv6 Timers

`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.
`Router(config-if)#ipv6 hello-interval eigrp 100 10`	Configures the hello interval for EIGRP for IPv6 process 100 to be 10 seconds. The default is 5 seconds.
`Router(config-if)#ipv6 hold-time eigrp 100 40`	Configures the hold timer for EIGRP for IPv6 process 100 to be 40 seconds. The default is 15 seconds.

EIGRPv6 Stub Routing

`Router(config)#ipv6 router eigrp 100`	Enters router configuration mode and creates an EIGRP IPv6 routing process
`Router(config-router)#eigrp stub`	Configures a router as a stub using EIGRP

NOTE: The same keywords in the **eigrp stub** command that work with EIGRP for IPv4 will also work with EIGRPv6: **connected | summary | static | redistributed | receive-only**.

Logging EIGRPv6 Neighbor Adjacency Changes

`Router(config)#ipv6 router eigrp 100`	Enters router configuration mode and creates an EIGRP IPv6 routing process.
`Router(config-router)#eigrp log-neighbor changes`	Enables the logging of changes in EIGRP for IPv6 neighbor adjacencies.
`Router(config-router)#eigrp log-neighbor-warnings 300`	Configures the logging intervals of EIGRP neighbor warning messages to 300 seconds. The default is 10 seconds.

Adjusting the EIGRPv6 Metric Weights

`Router(config)#ipv6 router eigrp 100`	Enters router configuration mode and creates an EIGRP IPv6 routing process.
`Router(config-router)#metric weights` *`tos k1 k2 k3 k4 k5`* `Router(config-router)#metric weights 0 1 1 1 1 1`	Changes the default *k* values used in metric calculation. These are the default values: tos=0, k1=1, k2=0, k3=1, k4=0, k5=0.

EIGRP Address Families

EIGRP supports multiple protocols and carries information about different route types. Named EIGRP configuration is hierarchical when displayed.

The two most commonly used address families are IPv4 unicast and IPv6 unicast. Multicast for both IPv4 and IPv6 is also supported. The default address families for both IPv4 and IPv6 are unicast.

`Router(config)#router eigrp TEST`	Creates a named EIGRP virtual instance called TEST.
	NOTE: The name of the virtual instance is locally significant only.
	NOTE: The name does not need to match between neighbor routers.
	NOTE: This command defines a single EIGRP instance that can be used for all address families. At least one address family must be defined.

`Router(config-router)#address-family ipv4 autonomous-system 1`	Enables the IPv4 address family and starts EIGRP autonomous system 1.
`Router(config-router-af)#network 172.16.10.0 0.0.0.255`	Enables EIGRP for IPv4 on interfaces in the 172.16.10.0 network.
`Router(config-router-af)#network 0.0.0.0`	Enables EIGRP for IPv4 on all IPv4 enabled interfaces.
	NOTE: In the **config-router-af** mode, you can define other general parameters for EIGRP, such as **router-id** or **eigrp stub**.
`Router(config-router-af)#af-interface gigabitethernet0/0`	Moves the router into the address family interface configuration mode for interface Gigabit Ethernet 0/0.
`Router(config-router-af-interface)#summary-address 192.168.10.0/23`	Configures a summary aggregate address.
`Router(config)#router eigrp TEST`	Creates a named EIGRP virtual instance called TEST.
`Router(config-router)#address-family ipv6 autonomous-system 1`	Enables the IPv6 address family and starts EIGRP autonomous system 1.
	NOTE: EIGRPv6 does not need to be configured on the interface. All IPv6 enabled interfaces are included in the EIGRPv6 process.
`Router(config-router-af)#af-interface default`	Moves the router into the address family interface configuration mode for all interfaces.
`Router(config-router-af-interface)#passive-interface`	Configures all IPv6 interfaces as passive for EIGRP.
`Router(config-router-af-interface)#exit`	Returns to router address family mode.
	NOTE: The complete command is **exit-af-interface**, but the more commonly used shortcut of **exit** is presented here.
`Router(config-router-af)#af-interface gigabitethernet0/0`	Moves the router into the address family interface configuration mode for interface Gigabit Ethernet 0/0.
`Router(config-router-af-interface)#no passive-interface`	Removes the passive interface configuration from this interface.

Named EIGRP Configuration Modes

Named EIGRP configuration mode gathers all EIGRP configurations in one place.

Mode	Commands Used in This Mode
Address-family configuration mode `Router(config-router-af)#`	General configuration commands: **eigrp stub** **network** **router-id**
Address-family interface configuration mode `Router(config-router-af-interface)#`	Interface-specific configuration commands: **hello-interval** **hold-time** **passive-interface** **summary-address**
Address-family topology configuration mode `Router(config-router-topology)#`	Configuration commands that affect the topology table: **maximum-paths** **redistribute** **variance**

Verifying EIGRP and EIGRPv6

`Router#clear ip route *`	Deletes all routes from the IPv4 routing table.
`Router#clear ip route 172.16.10.0`	Clears this specific route from the IPv4 routing table.
`Router#clear ipv6 route *`	Deletes all routes from the IPv6 routing table.
	NOTE: Clearing all routes from the routing table will cause high CPU utilization rates as the routing table is rebuilt.
`Router#clear ipv6 route 2001:db8:c18:3::/64`	Clears this specific route from the IPv6 routing table.
`Router#clear ipv6 traffic`	Resets IPv6 traffic counters.
`Router#show ip eigrp neighbors`	Displays the neighbor table.
`Router#show ip eigrp neighbors detail`	Displays a detailed neighbor table.
	TIP: The **show ip eigrp neighbors detail** command will verify whether a neighbor is configured as a stub router.
`Router#show ip eigrp interfaces`	Shows info for each interface.
`Router#show ip eigrp interface serial0/0/0`	Shows info for a specific interface.

`Router#show ip eigrp interface 100`	Shows info for interfaces running process 100.
`Router#show ip eigrp topology`	Displays the topology table.
	TIP: The **show ip eigrp topology** command shows you where your feasible successors are.
`Router#show ip eigrp topology all-links`	Displays all entries in the EIGRP topology table, including nonfeasible-successor sources.
`Router#show ip eigrp traffic`	Shows the number and type of packets sent and received.
`Router#show ip interface`	Displays the status of interfaces configured for IPv4.
`Router#show ip interface brief`	Displays a summarized status of interfaces configured for IPv4.
`Router#show ip protocols`	Shows the parameters and current state of the active routing protocol process.
`Router#show ip route`	Shows the complete routing table.
`Router#show ip route eigrp`	Shows a routing table with only EIGRP entries.
`Router#show ipv6 eigrp interfaces`	Displays IPv6 info for each interface.
`Router#show ipv6 eigrp interface serial 0/0/0`	Displays IPv6 info for specific interface.
`Router#show ipv6 eigrp interface 100`	Displays IPv6 info for interfaces running process 100.
`Router#show ipv6 eigrp neighbors`	Displays the EIGRPv6 neighbor table.
`Router#show ipv6 eigrp neighbors detail`	Displays a detailed EIGRPv6 neighbor table.
`Router#show ipv6 eigrp topology`	Displays the EIGRPv6 topology table.
`Router#show ipv6 interface`	Displays the status of interfaces configured for IPv6.
`Router#show ipv6 interface brief`	Displays a summarized status of interfaces configured for IPv6.
`Router#show ipv6 neighbors`	Displays IPv6 neighbor discovery cache information.
`Router#show ipv6 protocols`	Displays the parameters and current state of the active IPv6 routing protocol processes.
`Router#show ipv6 route`	Displays the current IPv6 routing table.
`Router#show ipv6 route eigrp`	Displays the current IPv6 routing table with only EIGRPv6 routes.
`Router#show ipv6 route summary`	Displays a summarized form of the current IPv6 routing table.

`Router#show ipv6 routers`	Displays IPv6 router advertisement information received from other routers.
`Router#show ipv6 traffic`	Displays statistics about IPv6 traffic.

Troubleshooting EIGRP

`Router#debug eigrp fsm`	Displays events/actions related to EIGRP feasible successor metrics (FSM).
`Router#debug eigrp packets`	Displays events/actions related to EIGRP packets.
`Router#debug eigrp neighbor`	Displays events/actions related to your EIGRP neighbors.
`Router#debug ip eigrp`	Displays events/actions related to EIGRP protocol packets.
`Router#debug ip eigrp notifications`	Displays EIGRP event notifications.
`Router#debug ipv6 eigrp`	Displays information about the EIGRP for IPv6 protocol.
`Router#debug ipv6 neighbor 2001:db8:c18:3::1`	Displays information about the specified EIGRP for IPv6 neighbor.
`Router#debug ipv6 neighbor notification`	Displays EIGRP for IPv6 events and notifications in the console of the router.
`Router#debug ipv6 neighbor summary`	Displays a summary of EIGRP for IPv6 routing information.
`Router#debug ipv6 packet`	Displays debug messages for IPv6 packets.
	TIP: Send your **debug** output to a syslog server to ensure that you have a copy of it in case your router is overloaded and needs to reboot.
`Router#debug ipv6 routing`	Displays debug messages for IPv6 routing table updates and route cache updates.

Configuration Example: EIGRPv4 and EIGRPv6 using Named Address Configuration

Figure 2-7 shows the network topology for the configuration that follows, which shows how to configure EIGRP using commands covered in this chapter.

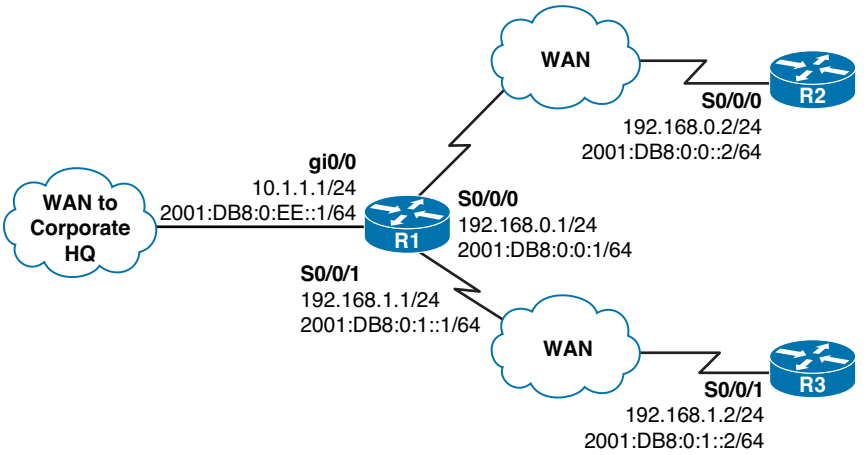


Figure 2-7 Network Topology for EIGRP Configuration

R1 Router

`R1>enable`	Enters privileged mode.
`R1#config t`	Moves to global configuration mode.
`R1(config)#router eigrp ConfigEG`	Creates a named EIGRP virtual instance called ConfigEG.
`R1(config-router)#address family ipv4 autonomous-system 1`	Enables the IPv4 address family and starts EIGRP autonomous system 1.
`R1(config-router-af)#network 10.1.1.0`	Enables EIGRP for IPv4 on interfaces in the 10.1.1.0 network.
`R1(config-router-af)#network 192.168.0.0`	Enables EIGRP for IPv4 on interfaces in the 192.168.0.0 network.
`R1(config-router-af)#network 192.168.1.0`	Enables EIGRP for IPv4 on interfaces in the 192.168.1.0 network.
`R1(config-router-af)#af-interface gigabitethernet0/0`	Moves the router into the address family interface configuration mode for interface Gigabit Ethernet 0/0.
`R1(config-router-af-interface)#summary-address 192.168.0.0/23`	Configures a summary aggregate address for the two serial prefixes.
`R1(config-router-af-interface)#exit`	Returns to address family configuration mode.
`R1(config-router-af)#exit`	Returns to EIGRP router configuration mode.
	NOTE: The complete command is **exit-address-family**.

`R1(config-router)#address-family ipv6 autonomous-system 1`	Enables the IPv6 address family and starts EIGRP autonomous system 1. All IPv6 enabled interfaces are included in the EIGRPv6 process.
`R1(config-router-af)#exit`	Returns to EIGRP router configuration mode.
`R1(config-router)#exit`	Returns to global configuration mode.
`R1(config)#exit`	Returns to privileged mode.
`R1#copy running-config startup-config`	Copies the running configuration to NVRAM.

R2 Router

`R2>enable`	Enters privileged mode.
`R2#config t`	Moves to global configuration mode.
`R2(config)#router eigrp ConfigEG`	Creates a named EIGRP virtual instance called ConfigEG.
`R2(config-router)#address family ipv4 autonomous-system 1`	Enables the IPv4 address family and starts EIGRP autonomous system 1.
`R2(config-router-af)#network 192.168.0.0`	Enables EIGRP for IPv4 on interfaces in the 192.168.0.0 network.
`R2(config-router-af)#exit`	Returns to EIGRP router configuration mode.
	NOTE: The complete command is **exit-address-family**.
`R2(config-router)#address-family ipv6 autonomous-system 1`	Enables the IPv6 address family and starts EIGRP autonomous system 1. All IPv6 enabled interfaces are included in the EIGRPv6 process.
`R2(config-router-af)#exit`	Returns to EIGRP router configuration mode.
`R2(config-router)#exit`	Returns to global configuration mode.
`R2(config)#exit`	Returns to privileged mode.
`R2#copy running-config startup-config`	Copies the running configuration to NVRAM.

R3 Router

`R3>enable`	Enters privileged mode.
`R3#config t`	Moves to global configuration mode.
`R3(config)#router eigrp ConfigEG`	Creates a named EIGRP virtual-instance called ConfigEG.

R3(config-router)#**address family ipv4 autonomous-system 1**	Enables the IPv4 address family and starts EIGRP autonomous system 1.
R3(config-router-af)#**network 192.168.1.0**	Enables EIGRP for IPv4 on interfaces in the 192.168.1.0 network.
R3(config-router-af)#**exit**	Returns to EIGRP router configuration mode.
	NOTE: The complete command is **exit-address-family**.
R3(config-router)#**address-family ipv6 autonomous-system 1**	Enables the IPv6 address family and starts EIGRP autonomous system 1. All IPv6 enabled interfaces are included in the EIGRPv6 process.
R3(config-router-af)#**exit**	Returns to EIGRP router configuration mode.
R3(config-router)#**exit**	Returns to global configuration mode.
R3(config)#**exit**	Returns to privileged mode.
R3#**copy running-config startup-config**	Copies the running configuration to NVRAM.

CHAPTER 3

Implementing a Scalable Multiarea Network OSPF-Based Solution

This chapter provides information about the following topics:

- OSPF message types
- LSA packet types
- Configuring OSPF
- Using wildcard masks with OSPF areas
- Configuring multiarea OSPF
- Loopback interfaces
- Router ID
- DR/BDR elections
- Passive interfaces
- Modifying cost metrics
- OSPF **auto-cost reference-bandwidth**
- OSPF LSDB overload protection
- Timers
- IP MTU
- Propagating a default route
- OSPF special area types
    - Stub areas
    - Totally stubby areas
    - Not-so-stubby areas
    - Totally NSSA
- Route summarization
    - Interarea route summarization
    - External route summarization
- Configuration example: virtual links
- OSPF and NBMA networks
- OSPF network types
    - Full-mesh Frame Relay: NBMA on physical interfaces
    - Full-mesh Frame Relay: broadcast on physical interfaces

        - Full-mesh Frame Relay: point-to-multipoint networks
        - Full-mesh Frame Relay: point-to-point networks on subinterfaces
    - OSPF over NBMA topology summary
    - IPv6 and OSPFv3
        - Enabling OSPF for IPv6 on an interface
        - OSPFv3 and stub/NSSA areas
        - Interarea OSPFv3 route summarization enabling an IPv4 router ID for OSPFv3
        - Forcing an SPF calculation
        - IPv6 on NBMA networks
        - OSPFv3 address families
        - Configuring the IPv6 address family in OSPFv3
            - Configuring the IPv4 address family in OSPFv3
            - Configuring parameters in address family mode
    - Verifying OSPF configuration
    - Troubleshooting OSPF
    - Configuration example: single-area OSPF
    - Configuration example: multiarea OSPF
    - Configuration example: OSPF and NBMA networks
    - Configuration example: OSPF and broadcast networks
    - Configuration example: OSPF and point-to-multipoint networks
    - Configuration example: OSPF and point-to-point networks using subinterfaces
    - Configuration example: IPv6 and OSPFv3

OSPF Message Types

Table 3-1 shows the different message types used by OSPF. Every OSPF packet is directly encapsulated in the IP header. The IP protocol number for OSPF is 89.

TABLE 3-1 OSPF Message Types

Type	Name	Description
1	Hello	Discovers neighbors and builds adjacencies between them
2	Database description (DBD)	Checks for database synchronization between routers
3	Link-state request (LSR)	Requests specific link-state advertisements (LSAs) from another router

Type	Name	Description
4	Link-state update (LSU)	Sends specifically requested LSAs
5	Link-state acknowledgment (LSAck)	Acknowledges the other packet types

OSPF LSA Types

Table 3-2 shows the different LSA types used by OSPF. LSAs are the building blocks of the OSPF link-state database (LSDB). Individually, LSAs act as database records. In combination, they describe the entire topology of an OSPF network area.

TABLE 3-2 OSPF LSA Types

Type	Name	Description
1	Router LSA	Describe the state of a router link to the area. Flooded within this single area.
2	Network LSA	Generated by designated routers (DRs) for multiaccess networks. Flooded within this single area.
3	Summary LSA	Used by an Area Border Router (ABR) to take information learned in one area and describes and summarizes it for another area.
4	ASBR summary LSA	Informs the rest of the OSPF domain how to reach the ASBR.
5	Autonomous system LSA	Generated by the ASBR, these LSAs describe routes to destinations external to the autonomous system.
6	Group membership LSA	Used in multicast OSPF (MOSPF) applications. MOSPF has been deprecated since OSPFv3 and is not currently used.
7	NSSA external link entry LSA	Used in special area type not-so-stubby-area (NSSA). Advertises external routes in an NSSA.
8	Link-local LSA for OSPFv3	Gives information about link-local addresses plus a list of IPv6 address on the link. Not supported by Cisco.
9	Opaque LSA	Reserved for future use.
10	Opaque LSA	Reserved for future use.
11	Opaque LSA	Reserved for future use.

Configuring OSPF

`Router(config)#router ospf 123`	Starts OSPF process 123. The process ID is any positive integer value between 1 and 65,535. The process ID is *not* related to the OSPF area. The process ID merely distinguishes one process from another within the device.
`Router(config-router)#network 172.16.10.0 0.0.0.255 area 0`	OSPF advertises interfaces, not networks. Uses the wildcard mask to determine which interfaces to advertise. Read this line to say, "Any interface with an address of 172.16.10.x is to run OSPF and be put into area 0."
	NOTE: The process ID number of one router does not have to match the process ID of any other router. Unlike Enhanced Interior Gateway Routing Protocol (EIGRP), matching this number across all routers does *not* ensure that network adjacencies will form.
`Router(config-router)#log-adjacency-changes detail`	Configures the router to send a syslog message when there is a change of state between OSPF neighbors.
	TIP: Although the **log-adjacency-changes** command is on by default, only up/down events are reported unless you use the **detail** keyword.

Using Wildcard Masks with OSPF Areas

When compared to an IP address, a wildcard mask will identify what addresses get matched to run OSPF and to be placed into an area:

- A 0 (zero) in a wildcard mask means to check the corresponding bit in the address for an exact match.
- A 1 (one) in a wildcard mask means to ignore the corresponding bit in the address—can be either 1 or 0.

Example 1: 172.16.0.0 0.0.255.255

172.16.0.0 = 10101100.00010000.00000000.00000000

0.0.255.255 = 00000000.00000000.11111111.11111111

Result = 10101100.00010000.*xxxxxxxx.xxxxxxxx*

172.16.*x.x* (Anything between 172.16.0.0 and 172.16.255.255 will match the example statement.)

TIP: An octet in the wildcard mask of all 0s means that the octet has to match the address exactly. An octet in the wildcard mask of all 1s means that the octet can be ignored.

Example 2: 172.16.8.0 0.0.7.255

172.168.8.0 = 10101100.00010000.00001000.00000000

0.0.0.7.255 = 00000000.00000000.00000111.11111111

result = 10101100.00010000.00001*xxx*.*xxxxxxxx*

00001*xxx* = 00001*000* to 00001*111* = 8 – 15

xxxxxxxx = 00000000 to 11111111 = 0 – 255

Anything between 172.16.8.0 and 172.16.15.255 will match the example statement.

`Router(config-router)#` **`network 172.16.10.1 0.0.0.0 area 0`**	Read this line to say, "Any interface with an exact address of 172.16.10.1 is to run OSPF and be put into area 0."
`Router(config-router)#` **`network 172.16.10.0 0.0.255.255 area 0`**	Read this line to say, "Any interface with an address of 172.16.*x*.*x* is to run OSPF and be put into area 0."
`Router(config-router)#` **`network 0.0.0.0 255.255.255.255 area 0`**	Read this line to say, "Any interface with any address is to run OSPF and be put into area 0."

Configuring Multiarea OSPF

`Router(config)#`**`router ospf 1`**	Starts OSPF process 1.
`Router(config-router)#`**`network 172.16.10.0 0.0.0.255 area 0`**	Read this line to say, "Any interface with an address of 172.16.10.*x* is to run OSPF and be put into area 0."
`Router(config-router)#`**`network 10.10.10.1 0.0.0.0 area 51`**	Read this line to say, "Any interface with an exact address of 10.10.10.1 is to run OSPF and be put into area 51."

Loopback Interfaces

`Router(config)#`**`interface loopback0`**	Creates a virtual interface named Loopback 0, and then moves the router to interface configuration mode.
`Router(config-if)#`**`ip address 192.168.100.1 255.255.255.255`**	Assigns the IP address to the interface.
	NOTE: Loopback interfaces are always "up and up" and do not go down unless manually shut down. This makes loopback interfaces great for use as an OSPF router ID.

Router ID

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#router-id 10.1.1.1`	Sets the router ID to 10.1.1.1. If this command is used on an OSPF router process that is already active (has neighbors), the new router ID is used at the next reload or at a manual OSPF process restart.
`Router(config-router)#no router-id 10.1.1.1`	Removes the static router ID from the configuration. If this command is used on an OSPF router process that is already active (has neighbors), the old router ID behavior is used at the next reload or at a manual OSPF process restart.

NOTE: To choose the router ID at the time of OSPF process initialization, the router uses the following criteria in this specific order:

Use the router ID specified in the **router-id** *ip address* command

Use the highest IP address of all active loopback interfaces on the router

Use the highest IP address among all active nonloopback interfaces

NOTE: To have the manually configured router ID take effect, you must clear the OSPF routing process with the **clear ip ospf process** command.

DR/BDR Elections

`Router(config)#interface fastethernet0/0`	Enters interface configuration mode.
`Router(config-if)#ip ospf priority 50`	Changes the OSPF interface priority to 50.
	NOTE: The assigned priority can be between 0 and 255. A priority of 0 makes the router ineligible to become a designated router (DR) or backup designated router (BDR). The highest priority wins the election and becomes the DR; the second highest priority becomes the BDR. A priority of 255 guarantees a tie in the election. If all routers have the same priority, regardless of the priority number, they tie. Ties are broken by the highest router ID. The default priority setting is 1.

Passive Interfaces

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#network 172.16.10.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.10.*x* is to be put into area 0."

`Router(config-router)#passive-interface fastethernet0/0`	Disables the sending of any OSPF packets on this interface.
`Router(config-router)#passive-interface default`	Disables the sending of any OSPF packets out all interfaces.
`Router(config-router)#no passive-interface serial 0/0/1`	Enables OSPF packets to be sent out interface serial 0/0/1, thereby allowing neighbor adjacencies to form.

Modifying Cost Metrics

`Router(config)#interface serial0/0/0`	Enters interface configuration mode.
`Router(config-if)#bandwidth 128`	If you change the bandwidth, OSPF will recalculate the cost of the link.
`Or`	
`Router(config-if)#ip ospf cost 1564`	Changes the cost to a value of 1564.
	NOTE: The cost of a link is determined by dividing the reference bandwidth by the interface bandwidth. The bandwidth of the interface is a number between 1 and 10,000,000. The unit of measurement is kilobits per second (Kbps).The cost is a number between 1 and 65,535. The cost has no unit of measurement; it is just a number.

OSPF auto-cost reference-bandwidth

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#auto-cost reference-bandwidth 1000`	Changes the reference bandwidth that OSPF uses to calculate the cost of an interface.
	NOTE: The range of the reference bandwidth is 1 to 4,294,967. The default is 100. The unit of measurement is megabits per second (Mbps).
	NOTE: The value set by the **ip ospf cost** command overrides the cost resulting from the **auto-cost** command.
	TIP: If you use the command **auto-cost reference-bandwidth** *reference-bandwidth*, you need to configure all the routers to use the same value. Failure to do so will result in routers using a different reference cost to calculate the shortest path, resulting in potential suboptimum routing paths.

OSPF LSDB Overload Protection

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-if)#max-lsa 12000`	Limits the number of non self-generated LSAs that this process can receive to 12,000. This number can be between 1 and 4,294,967,294.

NOTE: If other routers are configured incorrectly, causing, for example, a redistribution of a large number of prefixes, large numbers of LSAs can be generated. This can drain local CPU and memory resources. With the **max-lsa** *x* feature enabled, the router keeps count of the number of received (non-self-generated) LSAs that it keeps in its LSDB. An error message is logged when this number reaches a configured threshold number, and a notification is sent when it exceeds the threshold number.

If the LSA count still exceeds the threshold after 1 minute, the OSPF process takes down all adjacencies and clears the OSPF database. This is called the *ignore state*. In the ignore state, no OSPF packets are sent or received by interfaces that belong to the OSPF process. The OSPF process will remain in the ignore state for the time that is defined by the **ignore-time** parameter. If the OSPF process remains normal for the time that is defined by the **reset-time** parameter, the ignore state counter is reset to 0.

Timers

`Router(config-if)#ip ospf hello-interval timer 20`	Changes the hello interval timer to 20 seconds.
`Router(config-if)#ip ospf dead-interval 80`	Changes the dead interval timer to 80 seconds.
	NOTE: Hello and dead interval timers must match for routers to become neighbors.

NOTE: The default hello timer is 10 seconds on multiaccess and point-to-point segments. The default hello timer is 30 seconds on nonbroadcast multiaccess (NBMA) segments such as Frame Relay, X.25, or ATM.

NOTE: The default dead interval timer is 40 seconds on multiaccess and point-to-point segments. The default hello timer is 120 seconds on NBMA segments such as Frame Relay, X.25, or ATM.

NOTE: If you change the hello interval timer, the dead interval timer will automatically be adjusted to four times the new hello interval timer.

IP MTU

The IP maximum transmission unit (MTU) parameter determines the maximum size of a packet that can be forwarded without fragmentation.

`Router(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#ip mtu 1400`	Changes the MTU size to 1400 bytes. The range of this command is 68 to 1500 bytes.

CAUTION: The MTU size must match between all OSPF neighbors on a link. If OSPF routers have mismatched MTU sizes, they will not form a neighbor adjacency.

Propagating a Default Route

`Router(config)#ip route 0.0.0.0 0.0.0.0 serial0/0/0`	Creates a default route.
`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#default-information originate`	Sets the default route to be propagated to all OSPF routers.
`Router(config-router)#default-information originate always`	The **always** option will propagate a default "quad-0" route even if this router does not have a default route itself.
	NOTE: The **default-information originate** command or the **default-information originate always** command is usually only to be configured on your "entrance" or "gateway" router, the router that connects your network to the outside world—the Autonomous System Boundary Router (ASBR).

OSPF Special Area Types

This section covers four different special areas with respect to OSPF:

- Stub areas
- Totally stubby areas
- Not-so-stubby areas (NSSAs)
- Totally NSSA

Stub Areas

`ABR(config)#router ospf 1`	Starts OSPF process 1.
`ABR(config-router)#network 172.16.10.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.10.*x* is to run OSPF and be put into area 0."
`ABR(config-router)#network 172.16.20.0 0.0.0.255 area 51`	Read this line to say, "Any interface with an address of 172.16.20.x is to run OSPF and be put into area 51."
`ABR(config-router)#area 51 stub`	Defines area 51 as a stub area.
`ABR(config-router)#area 51 default-cost 10`	Defines the cost of a default route sent into the stub area. Default is 1.
	NOTE: This is an optional command.
`Internal(config)#router ospf 1`	Starts OSPF process 1.
`Internal(config-router)#network 172.16.20.0 0.0.0.255 area 51`	Read this line to say, "Any interface with an address of 172.16.20.*x* is to run OSPF and be put into area 51."
`Internal(config-router)#area 51 stub`	Defines area 51 as a stub area.
	NOTE: All routers in the stub area must be configured with the **area** *x* **stub** command, including the Area Border Router (ABR).

Totally Stubby Areas

`ABR(config)#router ospf 1`	Starts OSPF process 1.
`ABR(config-router)#network 172.16.10.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.10.*x* is to run OSPF and be put into area 0."
`ABR(config-router)#network 172.16.20.0 0.0.0.255 area 51`	Read this line to say, "Any interface with an address of 172.16.20.*x* is to run OSPF and be put into area 51."
`ABR(config-router)#area 51 stub no-summary`	Defines area 51 as a totally stubby area.
`Internal(config)#router ospf 1`	Starts OSPF process 1.
`Internal(config-router)#network 172.16.20.0 0.0.0.255 area 51`	Read this line to say, "Any interface with an address of 172.16.20.*x* is to run OSPF and be put into area 51."
`Internal(config-router)#area 51 stub`	Defines area 51 as a stub area.
	NOTE: Whereas all internal routers in the area are configured with the **area** *x* **stub** command, the ABR is configured with the **area** *x* **stub no-summary** command.

Not-So-Stubby Areas

`ABR(config)#router ospf 1`	Starts OSPF process 1.
`ABR(config-router)#network 172.16.10.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.10.*x* is to run OSPF and be put into area 0."
`ABR(config-router)#network 172.16.20.0 0.0.0.255 area 1`	Read this line to say, "Any interface with an address of 172.16.20.*x* is to run OSPF and be put into area 1."
`ABR(config-router)#area 1 nssa`	Defines area 1 as an NSSA.
`Internal(config)#router ospf 1`	Starts OSPF process 1.
`Internal(config-router)#network 172.16.20.0 0.0.0.255 area 1`	Read this line to say, "Any interface with an address of 172.16.20.*x* is to run OSPF and be put into area 1."
`Internal(config-router)#area 1 nssa`	Defines area 1 as an NSSA.
	NOTE: All routers in the NSSA stub area must be configured with the **area** *x* **nssa** command.

Totally NSSA

`ABR(config)#router ospf 1`	Starts OSPF process 1.
`ABR(config-router)#network 172.16.10.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.10.*x* is to run OSPF and be put into area 0."
`ABR(config-router)#network 172.16.20.0 0.0.0.255 area 11`	Read this line to say, "Any interface with an address of 172.16.20.*x* is to run OSPF and be put into area 11."
`ABR(config-router)#area 11 nssa no-summary`	Defines area 11 as a totally NSSA.
`Internal(config)#router ospf 1`	Starts OSPF process 1.
`Internal(config-router)#network 172.16.20.0 0.0.0.255 area 11`	Read this line to say, "Any interface with an address of 172.16.20.*x* is to run OSPF and be put into area 11."
`Internal(config-router)#area 11 nssa`	Defines area 11 as an NSSA.
	NOTE: Whereas all internal routers in the area are configured with the **area** *x* **nssa** command, the ABR is configured with the **area** *x* **nssa no-summary** command.

Route Summarization

In OSPF, there are two different types of summarization:

- Interarea route summarization
- External route summarization

The sections that follow provide the commands necessary to configure both types of summarization.

Interarea Route Summarization

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#area 1 range 192.168.64.0 255.255.224.0`	Summarizes area 1 routes to the specified summary address, before injecting them into a different area.
	NOTE: This command is to be configured on an ABR only.
	NOTE: By default, ABRs do *not* summarize routes between areas.

External Route Summarization

`Router(config)#router ospf 123`	Starts OSPF process 1.
`Router(config-router)#summary-address 192.168.64.0 255.255.224.0`	Advertises a single route for all the redistributed routes that are covered by a specified network address and netmask.
	NOTE: This command is to be configured on an ASBR only.
	NOTE: By default, ASBRs do *not* summarize routes.

Configuration Example: Virtual Links

Figure 3-1 shows the network topology for the configuration that follows, which demonstrates how to create a virtual link.

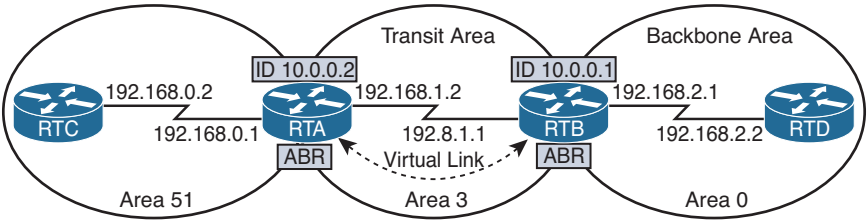


Figure 3-1 Virtual Areas: OSPF

`RTA(config)#router ospf 1`	Starts OSPF process 1.
`RTA(config-router)#router-id 10.0.0.2`	Sets the router ID to 10.0.0.2.
`RTA(config-router)#network 192.168.0.0 0.0.0.255 area 51`	Read this line to say, "Any interface with an address of 192.168.0.*x* is to run OSPF and be put into area 51."
`RTA(config-router)#network 192.168.1.0 0.0.0.255 area 3`	Read this line to say, "Any interface with an address of 192.168.1.*x* is to run OSPF and be put into area 3."
`RTA(config-router)#area 3 virtual-link 10.0.0.1`	Creates a virtual link with RTB.
`RTB(config)#router ospf 1`	Starts OSPF process 1.
`RTB(config-router)#router-id 10.0.0.1`	Sets the router ID to 10.0.0.1.
`RTB(config-router)#network 192.168.1.0 0.0.0.255 area 3`	Read this line to say, "Any interface with an address of 192.168.1.*x* is to run OSPF and be put into area 3."
`RTB(config-router)#network 192.168.2.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 192.168.2.*x* is to run OSPF and be put into area 0."
`RTB(config-router)#area 3 virtual-link 10.0.0.2`	Creates a virtual link with RTA.
	NOTE: A virtual link has the following two requirements: It must be established between two routers that share a common area and are both ABRs. One of these two routers must be connected to the backbone.
	NOTE: A virtual link is a temporary solution to a topology problem.
	NOTE: A virtual link cannot be configured through stub areas.
	NOTE: The routers establishing the virtual link do not have to be directly connected.

OSPF and NBMA Networks

OSPF is not well suited for nonbroadcast multiaccess (NBMA) networks such as Frame Relay or ATM. The term *multiaccess* means that an NBMA cloud is seen as a single network that has multiple devices attached to it, much like an Ethernet network. However, the *nonbroadcast* part of NBMA means that broadcast and multicast packets are not sent by default and that the devices use a pseudo broadcast for these types of packets. Therefore, a packet sent into this network might not be seen by all other routers, which differs from broadcast technologies such as Ethernet. OSPF will want to elect a DR and BDR because an NBMA network is multiaccess; however, because the network is also

nonbroadcast, there is no guarantee that all OSPF packets, such as Hello packets, would be received by other routers. This could affect the election of the DR because not all routers would know about all the other routers. The following sections list some possible solutions to dealing with OSPF in NBMA networks.

OSPF Network Types

OSPF network types can be described as either RFC compliant or Cisco proprietary:

- RFC compliant
  - NBMA
  - Point-to-multipoint
- Cisco proprietary
  - Point-to-multipoint nonbroadcast
  - Broadcast
  - Point-to-point

Full-Mesh Frame Relay: NBMA on Physical Interfaces

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#neighbor 10.1.1.2`	Identifies neighbor router.
`Router(config-router)#exit`	Returns to global configuration mode
`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.
`Router(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface.
`Router(config-if)#ip address 10.1.1.1 255.255.255.0`	Assigns an IP address and netmask to this interface.
`Router(config-if)#ip ospf network non-broadcast`	Defines OSPF nonbroadcast network type.
	NOTE: This is the default on physical interfaces.
`Router(config-if)#frame-relay map ip 10.1.1.2 100`	Maps the remote IP address 10.1.1.2 to data-link connection identifier (DLCI) 100.
`Router(config-if)#frame-relay map ip 10.1.1.3 200`	Maps the remote IP address 10.1.1.3 to DLCI 200.
	NOTE: Using the **neighbor** command will allow for an OSPF router to exchange routing information without multicasts and instead use unicasts to the manually entered neighbor IP address.

Full-Mesh Frame Relay: Broadcast on Physical Interfaces

`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.
`Router(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface.
`Router(config-if)#ip address 10.1.1.1 255.255.255.0`	Assigns an IP address and netmask to this interface.
`Router(config-if)#ip ospf network broadcast`	Changes the network type from the default nonbroadcast to broadcast.
`Router(config-if)#frame-relay map ip 10.1.1.2 100 broadcast`	Maps the remote IP address 10.1.1.2 to DLCI 100. Broadcast and multicast packets will now be forwarded.
`Router(config-if)#frame-relay map ip 10.1.1.3 200 broadcast`	Maps the remote IP address 10.1.1.3 to DLCI 200. Broadcast and multicast packets will now be forwarded.
`Router(config-if)#no shutdown`	Enables the interface.
`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#network 10.1.1.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 10.1.1.*x* is to run OSPF and be put into area 0."

Full-Mesh Frame Relay: Point-to-Multipoint Networks

NOTE: In this example, Inverse Address Resolution Protocol (ARP) is used to dynamically map IP addresses to DLCIs. Static maps could have been used, if desired.

NOTE: Point-to-multipoint networks treat private virtual circuits (PVCs) as a collection of point-to-point links rather than a multiaccess network. No DR/BDR election will take place.

NOTE: Point-to-multipoint networks might be your only alternative to broadcast networks in a multivendor environment.

NOTE: This design is an example of an RFC compliant network.

`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.
`Router(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface.
`Router(config-if)#ip address 10.1.1.1 255.255.255.0`	Assigns an IP address and netmask to this interface.
`Router(config-if)#ip ospf network point-to-multipoint`	Changes the network to a point-to-multipoint network.
`Router(config-if)#exit`	Returns to global configuration mode.

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#network 10.1.1.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 10.1.1.*x* is to run OSPF and be put into area 0."
`Router(config-router)#neighbor 10.1.1.2`	Identifies neighbor router.
`Router(config-router)#exit`	Returns to global configuration mode.
`Router(config)#interface serial0/0/1`	Moves to interface configuration mode.
`Router(config-if)#ip ospf network point-to-multipoint non-broadcast`	Creates a point-to-multipoint nonbroadcast mode.
	NOTE: Point-to-multipoint nonbroadcast mode is a Cisco extension to the RFC-compliant point-to-multipoint mode.
	NOTE: Neighbors must be manually defined in this mode.
	NOTE: DR/BDRs are not used in this mode.
	NOTE: Point-to-multipoint nonbroadcast mode is used in special cases where neighbors cannot be automatically discovered.

Full-Mesh Frame Relay: Point-to-Point Networks with Subinterfaces

`Router(config)#interface serial0/0/0`	Moves to interface configuration mode.
`Router(config-if)#encapsulation frame-relay`	Enables Frame Relay on this interface.
`Router(config-if)#no shutdown`	Enables the interface.
`Router(config-if)#interface serial0/0/0.300 point-to-point`	Creates subinterface 300 and makes it a point-to-point network. This is the default mode.
`Router(config-subif)#ip address 192.168.1.1 255.255.255.252`	Assigns an IP address and netmask.
`Router(config-subif)#frame-relay interface-dlci 300`	Assigns DLCI 300 to the subinterface.
`Router(config-subif)# interface serial0/0/0.400 point-to-point`	Creates subinterface 400 and makes it a point-to-point network.

`Router(config-subif)#ip address 192.168.1.5 255.255.255.252`	Assigns an IP address and netmask.
`Router(config-subif)#frame-relay interface-dlci 400`	Assigns DLCI 400 to the subinterface.
`Router(config-subif)#exit`	Returns to interface configuration mode.
`Router(config-if)#exit`	Returns to global configuration mode.
	NOTE: Point-to-point subinterfaces allow each PVC to be configured as a separate subnet. No DR/BDR election will take place with the default point-to-point mode.
	NOTE: The use of subinterfaces increases the amount of memory used on the router.

OSPF over NBMA Topology Summary

OSPF Mode	NBMA Preferred Topology	Subnet Address	Hello Timer	Adjacency	RFC or Cisco
Broadcast	Full or partial mesh	Same	10 seconds	Automatic, DR/BDR elected	Cisco
Nonbroadcast	Full or partial mesh	Same	30 seconds	Manual configuration, DR/BDR elected	RFC
Point-to-multipoint	Partial mesh or star	Same	30 seconds	Automatic, no DR/BDR	RFC
Point-to-multipoint nonbroadcast	Partial mesh or star	Same	30 seconds	Manual Configuration, no DR/BDR	Cisco
Point-to-point	Partial mesh or star, using subinterface	Different for each Subinterface	10 seconds	Automatic, no DR/BDR	Cisco

IPv6 and OSPFv3

Working with IPv6 requires modifications to any dynamic protocol. The current version of Open Shortest Path First (OSPF) Protocol, OSPFv2, was developed back in the late 1980s, when some parts of OSPF were designed to compensate for the inefficiencies of routers at that time. Now that router technology has dramatically increased, rather than modify OSPFv2 for IPv6, it was decided to create a new version of OSPF (OSPFv3) not just for IPv6, but for other, newer technologies, too. This section covers using IPv6 with OSPFv3.

Enabling OSPF for IPv6 on an Interface

`Router(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router.
	NOTE: This command is required before any IPv6 routing protocol can be configured.
`Router(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#ipv6 address 2001:db8:0:1::/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
`Router(config-if)#ipv6 ospf 1 area 0`	Enables OSPFv3 process 1 on the interface and places this interface into area 0.
	NOTE: The OSPFv3 process is created automatically when OSPFv3 is enabled on an interface.
	NOTE: The **ipv6 ospf** *x* **area** *y* command has to be configured on each interface that will take part in OSPFv3.
	NOTE: If a router ID has not been created first, the router will return a warning stating that the process could not pick a router ID. It will then tell you to manually configure a router ID.
`Router(config-if)#ipv6 ospf priority 30`	Assigns a priority number to this interface for use in the designated router (DR) election. The priority can be a number from 0 to 255. The default is 1. A router with a priority set to 0 is ineligible to become the DR or the backup DR (BDR).
`Router(config-if)#ipv6 ospf cost 20`	Assigns a cost value of 20 to this interface. The cost value can be an integer value from 1 to 65,535.
`Router(config-if)#ipv6 ospf neighbor FE80::A8BB:CCFF:FE00:C01`	Configures a neighbor. For use on NBMA networks.
`Router(config)#ospfv3 1 ipv6 area 0`	Enables OSPFv3 instance 1 with the IPv6 address family in area 0.
`Router(config)#ospfv3 1 ipv4 area 0`	Enables OSPFv3 instance 1 with the IPv4 address family in area 0.

OSPFv3 and Stub/NSSA Areas

`Router(config)#ipv6 router ospf`	Creates the OSPFv3 process if it has not already been created, and moves to router configuration mode.
`Router(config-rtr)#area 1 stub`	The router is configured to be part of a stub area.

`Router(config-rtr)#area 1 stub no-summary`	The router is configured to be in a totally stubby area. Only the ABR requires this **no-summary** keyword.
`Router(config-rtr)#area 1 nssa`	The router is configured to be in an NSSA.
`Router(config-rtr)#area 1 nssa no summary`	The router is configured to be in a totally stubby, NSSA area. Only the ABR requires the **no-summary** keyword.

Interarea OSPFv3 Route Summarization

`Router(config)#ipv6 router ospf 1`	Creates the OSPFv3 process if it has not already been created, and moves to router configuration mode
`Router(config-rtr)#area 1 range 2001:db8::/48`	Summarizes area 1 routes to the specified summary address, at an area boundary, before injecting them into a different area

Enabling an IPv4 Router ID for OSPFv3

`Router(config)#ipv6 router ospf 1`	Creates the OSPFv3 process if it has not already been created, and moves to router configuration mode.
`Router(config-rtr)#router-id 192.168.254.255`	Creates an IPv4 32-bit router ID for this router.
	NOTE: In OSPFv3 for IPv6, it is possible that no IPv4 addresses will be configured on any interface. In this case, the user must use the **router-id** command to configure a router ID before the OSPFv3 process will be started. If an IPv4 address does exist when OSPFv3 for IPv6 is enabled on an interface, that IPv4 address is used for the router ID. If more than one IPv4 address is available, a router ID is chosen using the same rules as for OSPF Version 2.

Forcing an SPF Calculation

`Router#clear ipv6 ospf 1 process`	The OSPF database is cleared and repopulated, and then the SPF algorithm is performed.
`Router#clear ipv6 ospf 1 force-spf`	The OSPF database is not cleared; just an SPF calculation is performed.

CAUTION: As with OSPFv2, clearing the OSPFv3 database and forcing a recalculation of the shortest path first (SPF) algorithm is processor intensive and should be used with caution.

IPv6 on NBMA Networks

The behavior of IPv6 unicast forwarding on Frame Relay networks is the same as IPv4 unicast forwarding. There are, however, two big differences when configuring IPv6 for unicast forwarding:

- You must configure mappings for link-local addresses because they will often be used by control plane operations such as routing protocols. The link-local address is used as the next-hop address for any routes installed in the routing table by an Interior Gateway Protocol. If the next-hop link-local address is not reachable because it is not mapped to the correct DLCI, the remote network will be unreachable. Use the **frame-relay map ipv6** command in interface configuration mode to achieve this.
- You must explicitly enable IPv6 unicast routing using the **ipv6 unicast-routing** global configuration command before any IPv6 routing protocol can be configured and before any IPv6 routing can occur.

OSPFv3 Address Families

The OSPFv3 address families feature is supported as of Cisco IOS Release 15.1(3)S and Cisco IOS Release 15.2(1)T. Cisco devices that run software older than these releases and third-party devices will not form neighbor relationships with devices running the address family feature for the IPv4 address family because they do not set the address family bit. Therefore, those devices will not participate in the IPv4 address family SPF calculations and will not install the IPv4 OSPFv3 routes in the IPv6 RIB.

NOTE: Devices running OSPFv2 will not communicate with devices running OSPFv3 for IPv4.

NOTE: To use the IPv4 unicast address families (AFs) in OSPFv3, you must enable IPv6 on a link, although the link may not be participating in IPv6 unicast AF.

NOTE: With the OSPFv3 address families feature, users may have two processes per interface, but only one process per AF. If the AF is IPv4, an IPv4 address must first be configured on the interface, but IPv6 must be enabled on the interface.

Configuring the IPv6 Address Family in OSPFv3

`Router(config)#router ospfv3 1`	Enables OSPFv3 router configuration mode for the IPv4 or IPv6 address family.
`Router(config-router)#address-family ipv6 unicast` `Router(config-router-af)#`	Enters IPv6 address family configuration mode for OSPFv3. Notice the prompt change.

Configuring the IPv4 Address Family in OSPFv3

`Router(config)#router ospfv3 1`	Enables OSPFv3 router configuration mode for the IPv4 or IPv6 address family.
`Router(config-router)#address-family ipv4 unicast` `Router(config-router-af)#`	Enters IPv4 address family configuration mode for OSPFv3. Notice the prompt change.

Configuring Parameters in Address Family Mode

`Router(config-router-af)#area 1 range 2001:DB8:0:0::0/128`	Summarizes area 1 routes to the specified summary address, at an area boundary, before injecting them into a different area.
`Router(config-router-af)#default area 1`	Resets OSPFv3 area 1 parameter to their default values.
`Router(config-router-af)#area 0 range 172.16.0.0 255.255.0.0`	Summarizes area 0 routes to specified summary address, before injecting them into a different area.
`Router(config-router-af)#default-metric 10`	Sets default metric values for IPv4 and IPv6 routes redistributed into the OSPFv3 routing protocol.
`Router(config-router-af)#maximum-paths 4`	Sets the maximum number of equal-cost routes that a process for OSPFv3 routing can support.
`Router(config-router-af)#summary-prefix FEC0::/24`	Configures an IPv6 summary prefix. This is done on an ASBR.
	NOTE: Other commands that are available in AF mode include the following: **passive-interface** **router-id** **area stubnssa stub**

Verifying OSPF Configuration

`Router#show ip protocol`	Displays parameters for all protocols running on the router.
`Router#show ip route`	Displays a complete IP routing table.
`Router#show ip route ospf`	Displays the OSPF routes in the routing table.
`Router#show ip route ospfv3`	Displays the OSPFv3 routes in the routing table.

`Router#show ip ospf`	Displays basic information about OSPF routing processes.
`Router#show ip ospf border-routers`	Displays border and boundary router information.
`Router#show ip ospf database`	Displays the contents of the OSPF database.
`Router#show ip ospf database asbr-summary`	Displays type 4 LSAs.
`Router#show ip ospf database external`	Displays type 5 LSAs.
`Router#show ip ospf database nssa-external`	Displays NSSA external link states.
`Router#show ip ospf database network`	Displays network LSAs.
`Router#show ip ospf database router self-originate`	Displays locally generated LSAs.
`Router#show ip ospf database summary`	Displays a summary of the OSPF database.
`Router#show ip ospf interface`	Displays OSPF info as it relates to all interfaces.
`Router#show ip ospf interface fastethernet0/0`	Displays OSPF information for interface fastethernet 0/0.
`Router#show ip ospf neighbor`	Lists all OSPF neighbors and their states.
`Router#show ip ospf neighbor detail`	Displays a detailed list of neighbors.
`Router#show ipv6 interface`	Displays the status of interfaces configured for IPv6.
`Router#show ipv6 interface brief`	Displays a summarized status of interfaces configured for IPv6.
`Router#show ipv6 neighbors`	Displays IPv6 neighbor discovery cache information.
`Router#show ipv6 ospf`	Displays general information about the OSPFv3 routing process.
`Router#show ipv6 ospf border-routers`	Displays the internal OSPF routing table entries to an ABR or ASBR.
`Router#show ipv6 ospf database`	Displays OSPFv3-related database information.
`Router#show ipv6 ospf database database-summary`	Displays how many of each type of LSA exist for each area in the database.
`Router#show ipv6 ospf interface`	Displays OSPFv3-related interface information.
`Router#show ipv6 ospf neighbor`	Displays OSPFv3-related neighbor information.

`Router#show ipv6 ospf virtual-links`	Displays parameters and the current state of OSPFv3 virtual links.
`Router#show ipv6 protocols`	Displays the parameters and current state of the active IPv6 routing protocol processes.
`Router#show ipv6 route`	Displays the current IPv6 routing table.
`Router#show ipv6 route summary`	Displays a summarized form of the current IPv6 routing table.
`Router#show ipv6 routers`	Displays IPv6 router advertisement information received from other routers.
`Router#show ipv6 traffic`	Displays statistics about IPv6 traffic.
`Router#show ip ospf virtual-links`	Displays information about virtual links.
`Router#show ospfv3 database`	Displays the OSPFv3 database.
`Router#show ospfv3 nighbor`	Displays OSPFv3 neighbor information on a per-interface basis.

Troubleshooting OSPF

`Router#clear ip route *`	Clears the entire routing table, forcing it to rebuild.
`Router#clear ip route a.b.c.d`	Clears a specific route to network a.b.c.d.
`Router#clear ipv6 route *`	Deletes all routes from the IPv6 routing table.
`Router#clear ipv6 route 2001:db8:c18:3::/64`	Clears this specific route from the IPv6 routing table.
`Router#clear ipv6 traffic`	Resets IPv6 traffic counters.
`Router#clear ip ospf counters`	Resets OSPF counters.
`Router#clear ip ospf process`	Resets the *entire* OSPF process, forcing OSPF to re-create neighbors, database, and routing table.
`Router#clear ip ospf 3 process`	Resets OSPF process 3, forcing OSPF to re-create neighbors, database, and routing table.
`Router#clear ipv6 ospf process`	Resets the *entire* OSPFv3 process, forcing OSPFv3 to re-create neighbors, database, and routing table.
`Router#clear ipv6 ospf 3 process`	Resets OSPFv3 process 3, forcing OSPF to re-create neighbors, database, and routing table.
`Router#debug ip ospf events`	Displays *all* OSPF events.
`Router#debug ip ospf adj`	Displays various OSPF states and DR/BDR election between adjacent routers.
`Router#debug ipv6 ospf adj`	Displays debug messages about the OSPF adjacency process.

`Router#debug ipv6 packet`	Displays debug messages for IPv6 packets.
`Router#debug ip ospf packets`	Displays OSPF packets.
`Router#debug ipv6 routing`	Displays debug messages for IPv6 routing table updates and route cache updates.
`Router#undebug all`	Turns off all **debug** commands.

Configuration Example: Single-Area OSPF

Figure 3-2 shows the network topology for the configuration that follows, which demonstrates how to configure single-area OSPF using the commands covered in this chapter.

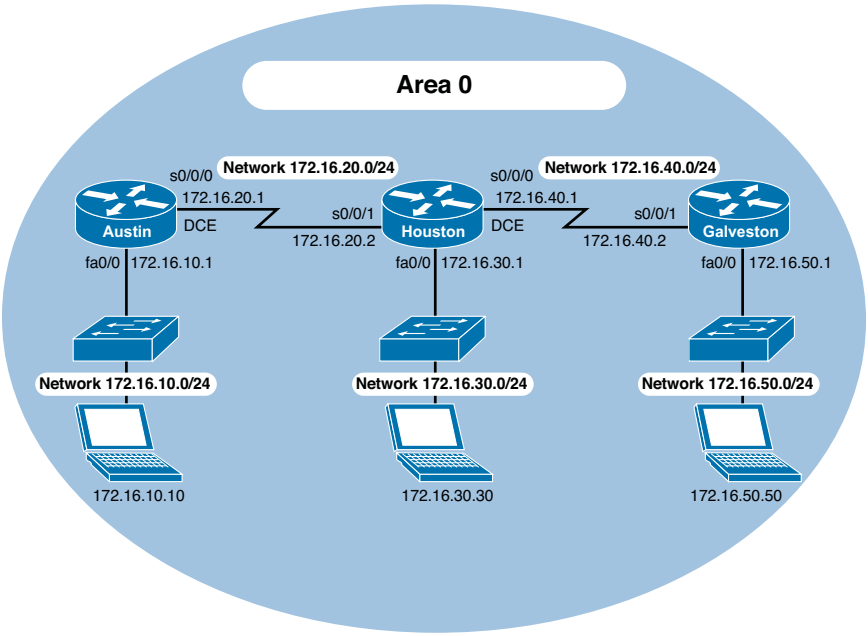


Figure 3-2 Network Topology for Single-Area OSPF Configuration

Austin Router

`Austin(config)#router ospf 1`	Starts OSPF process 1.
`Austin(config-router)#network 172.16.10.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.10.*x* is to run OSPF and be put into area 0."
`Austin(config-router)#network 172.16.20.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.20.*x* is to run OSPF and be put into area 0."

`Austin(config-router)#` **`<CTRL> z`**	Returns to privileged mode.
`Austin#`**`copy running-config startup-config`**	Saves the configuration to NVRAM.

Houston Router

`Houston(config)#`**`router ospf 1`**	Starts OSPF process 1.
`Houston(config-router)#`**`network 172.16.0.0 0.0.255.255 area 0`**	Read this line to say, "Any interface with an address of 172.16.*x.x* is to run OSPF and be put into area 0." One statement will now advertise all three interfaces.
`Houston(config-router)#`**`<CTRL> z`**	Returns to privileged mode.
`Houston#`**`copy running-config startup-config`**	Saves the configuration to NVRAM.

Galveston Router

`Galveston(config)#`**`router ospf 1`**	Starts OSPF process 1.
`Galveston(config-router)#`**`network 172.16.40.2 0.0.0.0 area 0`**	Any interface with an exact address of 172.16.40.2 is to run OSPF and be put into area 0. This is the most precise way to place an exact address into the OSPF routing process.
`Galveston(config-router)#`**`network 172.16.50.1 0.0.0.0 area 0`**	Read this line to say, "Any interface with an exact address of 172.16.50.1 is to be put into area 0."
`Galveston(config-router)#`**`<CTRL> z`**	Returns to privileged mode.
`Galveston#`**`copy running-config startup-config`**	Saves the configuration to NVRAM.

Configuration Example: Multiarea OSPF

Figure 3-3 shows the network topology for the configuration that follows, which demonstrates how to configure multiarea OSPF using the commands covered in this chapter.

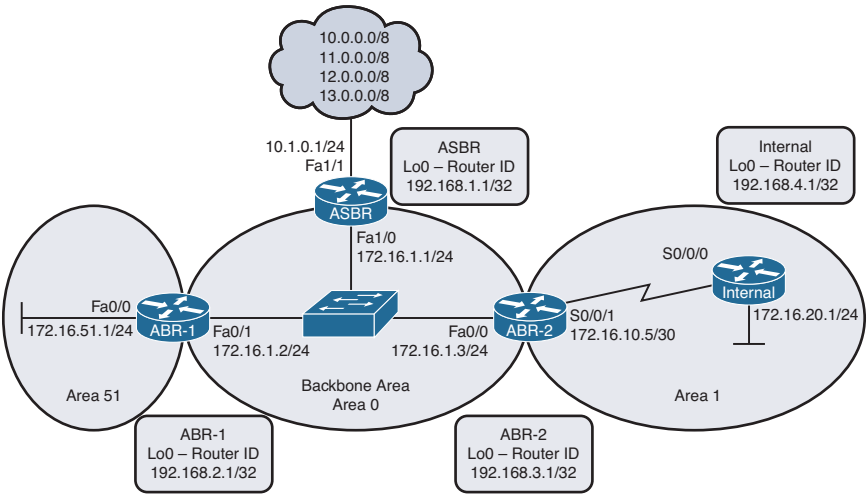


Figure 3-3 Network Topology for Multiarea OSPF Configuration

ASBR Router

`Router> enable`	Moves to privileged mode.
`Router#configure terminal`	Moves to global configuration mode.
`Router(config)#hostname ASBR`	Sets the router hostname.
`ASBR(config)#interface loopback0`	Enters loopback interface mode.
`ASBR(config-if)#ip address 192.168.1.1 255.255.255.255`	Assigns an IP address and netmask.
`ASBR(config-if)#description Router ID`	Sets a locally significant description.
`ASBR(config-if)#exit`	Returns to global configuration mode.
`ASBR(config)#ip route 0.0.0.0 0.0.0.0 10.1.0.2 fastethernet0/1`	Creates default route. Using both an exit interface and next-hop address on a Fast Ethernet interface prevents recursive look-ups in the routing table.
`ASBR(config)#ip route 11.0.0.0 0.0.0.0 null0`	Creates a static route to a null interface. In this example, these routes represent a simulated remote destination.
`ASBR(config)#ip route 12.0.0.0 0.0.0.0 null0`	Creates a static route to a null interface. In this example, these routes represent a simulated remote destination.
`ASBR(config)#ip route 13.0.0.0 0.0.0.0 null0`	Creates a static route to a null interface. In this example, these routes represent a simulated remote destination.
`ASBR(config)#router ospf 1`	Starts OSPF process 1.

`ASBR(config-router)#network 172.16.1.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.1.*x* is to run OSPF and be put into area 0."
`ASBR(config-router)#default-information originate`	Sets the default route to be propagated to all OSPF routers.
`ASBR(config-router)#redistribute static`	Redistributes static routes into the OSPF process. This turns the router into an ASBR because static routes are not part of OSPF, and the definition of an ASBR is a router that sits between OSPF and another routing process—in this case, static routing.
`ASBR(config-router)#exit`	Returns to global configuration mode.
`ASBR(config)#exit`	Returns to privileged mode.
`ASBR#copy running-config startup-config`	Saves the configuration to NVRAM.

ABR-1 Router

`Router> enable`	Moves to privileged mode.
`Router#configure terminal`	Moves to global configuration mode.
`Router(config)#hostname ABR-1`	Sets the router hostname.
`ABR-1(config)#interface loopback0`	Enters loopback interface mode.
`ABR-1(config-if)#ip address 192.168.2.1 255.255.255.255`	Assigns an IP address and netmask.
`ABR-1(config-if)#description Router ID`	Sets a locally significant description.
`ABR-1(config-if)#exit`	Returns to global configuration mode.
`ABR-1(config)#interface fastethernet0/1`	Enters interface configuration mode.
`ABR-1(config-if)#ip ospf priority 200`	Sets the priority for the DR/BDR election process. This router will win and become the DR.
`ABR-1(config-if)#no shutdown`	Enables the interface.
`ABR-1(config-if)#exit`	Returns to global configuration mode.
`ABR-1(config)#router ospf 1`	Starts OSPF process 1.
`ABR-1(config-router)#network 172.16.1.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.1.*x* is to run OSPF and be put into area 0."
`ABR-1(config-router)#network 172.16.51.1 0.0.0.0 area 51`	Read this line to say, "Any interface with an exact address of 172.16.51.1 is to run OSPF and be put into area 51."
`ABR-1(config-router)#exit`	Returns to global configuration mode.

`ABR-1(config)#exit`	Returns to privileged mode.
`ABR-1(config)#copy running-config startup-config`	Saves the configuration to NVRAM.

ABR-2 Router

`Router>enable`	Moves to privileged mode.
`Router#configure terminal`	Moves to global configuration mode.
`Router(config)#hostname ABR-2`	Sets the router hostname.
`ABR-2(config)#interface loopback0`	Enters loopback interface mode.
`ABR-2(config-if)#ip address 192.168.3.1 255.255.255.255`	Assigns an IP address and netmask.
`ABR-2(config-if)#description Router ID`	Sets a locally significant description.
`ABR-2(config-if)#exit`	Returns to global configuration mode.
`ABR-2(config)#interface fastethernet0/0`	Enters interface configuration mode.
`ABR-2(config-if)#ip ospf priority 100`	Sets the priority for the DR/BDR election process. This router will become the BDR to ABR-1's DR.
`ABR-2(config-if)#no shutdown`	Enables the interface.
`ABR-2(config-if)#exit`	Returns to global configuration mode.
`ABR-2(config)#router ospf 1`	Starts OSPF process 1.
`ABR-2(config-router)#network 172.16.1.0 0.0.0.255 area 0`	Read this line to say, "Any interface with an address of 172.16.1.*x* is to run OSPF and be put into area 0."
`ABR-2(config-router)#network 172.16.10.4 0.0.0.3 area 1`	Read this line to say "Any interface with an address of 172.16.10.4–7 is to run OSPF and be put into area 1."
`ABR-2(config-router)#area 1 stub`	Makes area 1 a stub area. LSA type 4 and type 5s are blocked and not sent into area 1. A default route is injected into the stub area, pointing to the ABR.
`ABR-2(config-router)#exit`	Returns to global configuration mode.
`ABR-2(config)#exit`	Returns to privileged mode.
`ABR-2(config)#copy running-config startup-config`	Saves the configuration to NVRAM.

Internal Router

`Router>enable`	Moves to privileged mode.
`Router#configure terminal`	Moves to global configuration mode.

`Router(config)#hostname Internal`	Sets the router hostname.
`Internal(config)#interface loopback0`	Enters loopback interface mode.
`Internal(config-if)#ip address 192.168.4.1 255.255.255.255`	Assigns an IP address and netmask.
`Internal(config-if)#description Router ID`	Sets a locally significant description.
`Internal(config-if)#exit`	Returns to global configuration mode.
`Internal(config)#router ospf 1`	Starts OSPF process 1.
`Internal(config-router)#network 172.16.0.0 0.0.255.255 area 1`	Read this line to say, "Any interface with an address of 172.16.*x.x* is to run OSPF and be put into area 1."
`Internal(config-router)#area 1 stub`	Makes area 1 a stub area.
`Internal(config-router)#exit`	Returns to global configuration mode.
`Internal(config)#exit`	Returns to privileged mode.
`Internal(config)#copy running-config startup-config`	Saves the configuration to NVRAM.

Configuration Example: OSPF and NBMA Networks

Figure 3-4 shows the network topology for the configuration that follows, which demonstrates how to configure OSPF on an NBMA network using the commands covered in this chapter.

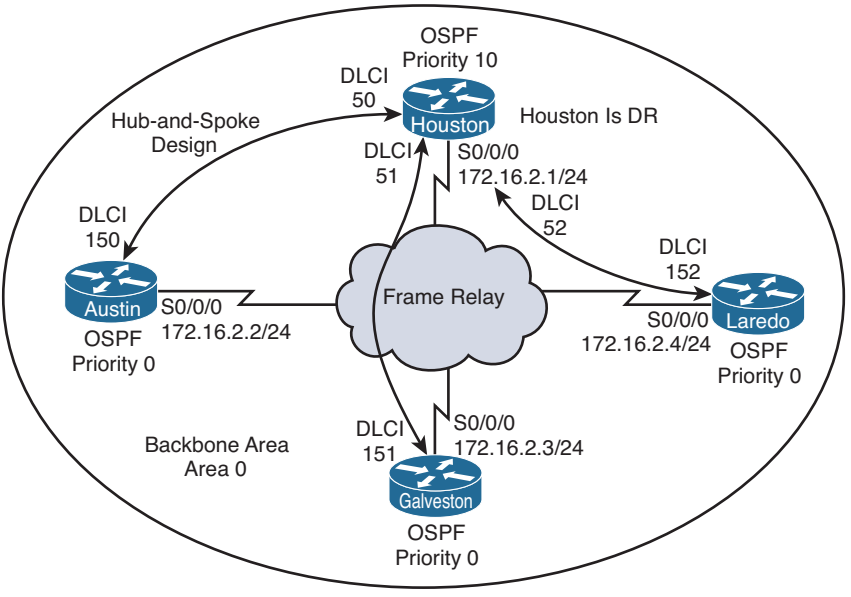


Figure 3-4 Network Topology for OSPF Configuration on an NBMA Network

Houston Router

`Houston(config)#interface serial0/0/0`	Enters interface configuration mode.
`Houston(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Houston(config-if)#ip address 172.16.2.1 255.255.255.0`	Assigns an IP address and netmask.
`Houston(config-if)#frame-relay map ip 172.16.2.2 50`	Maps the remote IP address to local DLCI 50.
`Houston(config-if)#frame-relay map ip 172.16.2.3 51`	Maps the remote IP address to local DLCI 51.
`Houston(config-if)#frame-relay map ip 172.16.2.4 52`	Maps the remote IP address to local DLCI 52.
`Houston(config-if)#ip ospf priority 10`	Changes the OSPF interface priority to 10.
`Houston(config-if)#no shutdown`	Enables the interface.
`Houston(config-if)#exit`	Returns to global configuration mode.
`Houston(config)#router ospf 1`	Starts OSPF process 1.
`Houston(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Houston(config-router)#neighbor 172.16.2.2`	Identifies neighbor (Austin) to Houston.
`Houston(config-router)#neighbor 172.16.2.3`	Identifies neighbor (Galveston) to Houston.
`Houston(config-router)#neighbor 172.16.2.4`	Identifies neighbor (Laredo) to Houston.
`Houston(config-router)#exit`	Returns to global configuration mode.
`Houston(config)#exit`	Returns to privileged mode.
`Houston#copy running-config startup-config`	Saves the configuration to NVRAM.

Austin Router

`Austin(config)#interface serial0/0/0`	Enters interface configuration mode.
`Austin(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Austin(config-if)#ip address 172.16.2.2 255.255.255.0`	Assigns an IP address and netmask.
`Austin(config-if)#frame-relay map ip 172.16.2.1 150`	Maps the remote IP address to local DLCI 150.

`Austin(config-if)#frame-relay map ip 172.16.2.3 150`	Maps the remote IP address to local DLCI 150.
`Austin(config-if)#frame-relay map ip 172.16.2.4 150`	Maps the remote IP address to local DLCI 150.
`Austin(config-if)#ip ospf priority 0`	Changes the OSPF interface priority to 0.
`Austin(config-if)#no shutdown`	Enables the interface.
`Austin(config-if)#exit`	Returns to global configuration mode.
`Austin(config)#router ospf 1`	Starts OSPF process 1.
`Austin(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Austin(config-router)#exit`	Returns to global configuration mode.
`Austin(config)#exit`	Returns to privileged mode.
`Austin#copy running-config startup-config`	Saves the configuration to NVRAM.

Galveston Router

`Galveston(config)#interface serial0/0/0`	Enters interface configuration mode.
`Galveston(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Galveston(config-if)#ip address 172.16.2.3 255.255.255.0`	Assigns an IP address and netmask.
`Galveston(config-if)#frame-relay map ip 172.16.2.1 151`	Maps the remote IP address to local DLCI 151. Note that the **broadcast** keyword is not used here. Broadcast and multicasts will not be forwarded.
`Galveston(config-if)#frame-relay map ip 172.16.2.2 151`	Maps the remote IP address to local DLCI 151.
`Galveston(config-if)#frame-relay map ip 172.16.2.4 151`	Maps the remote IP address to local DLCI 151.
`Galveston(config-if)#ip ospf priority 0`	Changes the OSPF interface priority to 0.
`Galveston(config-if)#no shutdown`	Enables the interface.
`Galveston(config-if)#exit`	Returns to global configuration mode.
`Galveston(config)#router ospf 1`	Starts OSPF process 1.
`Galveston(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Austin(config-if)#ip ospf priority 0`	Changes the OSPF interface priority to 0.

`Galveston(config-router)#exit`	Returns to global configuration mode.
`Galveston(config)#exit`	Returns to privileged mode.
`Galveston#copy running-config startup-config`	Saves the configuration to NVRAM.

Laredo Router

`Laredo(config)#interface serial0/0/0`	Enters interface configuration mode.
`Laredo(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Laredo(config-if)#ip address 172.16.2.4 255.255.255.0`	Assigns an IP address and netmask.
`Laredo(config-if)#frame-relay map ip 172.16.2.1 152`	Maps the remote IP address to local DLCI 152.
`Laredo(config-if)#frame-relay map ip 172.16.2.2 152`	Maps the remote IP address to local DLCI 152.
`Laredo(config-if)#frame-relay map ip 172.16.2.3 152`	Maps the remote IP address to local DLCI 152.
`Laredo(config-if)#ip ospf priority 0`	Changes the OSPF interface priority to 0.
`Laredo(config-if)#no shutdown`	Enables the interface.
`Laredo(config-if)#exit`	Returns to global configuration mode.
`Laredo(config)#router ospf 1`	Starts OSPF process 1.
`Laredo(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Laredo(config-router)#exit`	Returns to global configuration mode.
`Laredo(config)#exit`	Returns to privileged mode.
`Laredo#copy running-config startup-config`	Saves the configuration to NVRAM.

Configuration Example: OSPF and Broadcast Networks

Figure 3-5 shows the network topology for the configuration that follows, which demonstrates how to configure OSPF on a broadcast network using the commands covered in this chapter.

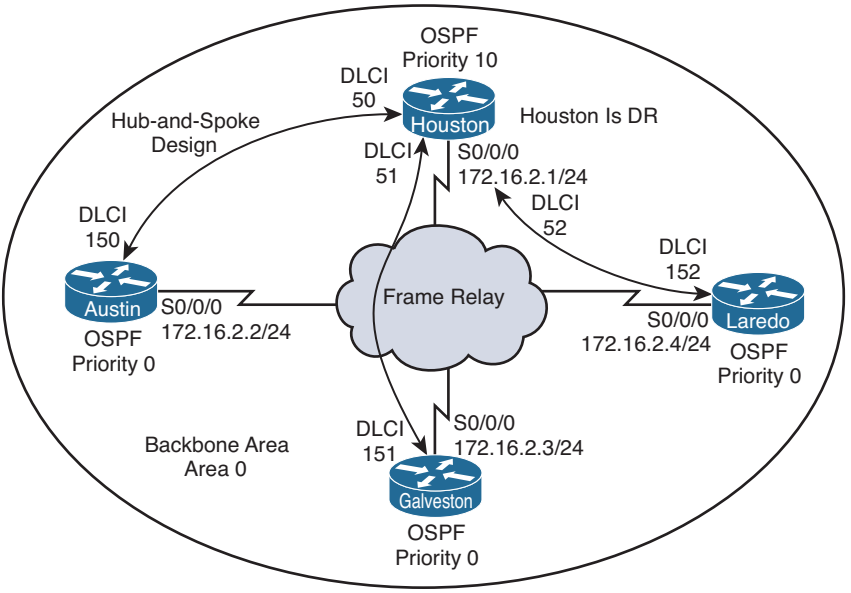


Figure 3-5 Network Topology for OSPF Configuration on a Broadcast Network

Houston Router

`Houston(config)#interface serial0/0/0`	Enters interface configuration mode.
`Houston(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Houston(config-if)#ip address 172.16.2.1 255.255.255.0`	Assigns an IP address and netmask.
`Houston(config-if)#ip ospf network broadcast`	Changes the network type from the default nonbroadcast to broadcast.
`Houston(config-if)#ip ospf priority 10`	Sets the priority to 10 for the DR/BDR election process.
`Houston(config-if)#frame-relay map ip 172.16.2.2 50 broadcast`	Maps the remote IP address to local DLCI 50. Broadcast and multicasts will now be forwarded.
`Houston(config-if)#frame-relay map ip 172.16.2.3 51 broadcast`	Maps the remote IP address to local DLCI 51. Broadcast and multicasts will now be forwarded.
`Houston(config-if)#frame-relay map ip 172.16.2.4 52 broadcast`	Maps the remote IP address to local DLCI 52. Broadcast and multicasts will now be forwarded.
`Houston(config-if)#no shut`	Enables the interface.
`Houston(config-if)#exit`	Returns to global configuration mode.

`Houston(config)#router ospf 1`	Starts OSPF process 1.
`Houston(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Houston(config-router)#exit`	Returns to global configuration mode.
`Houston(config)#exit`	Returns to privileged mode.
`Houston#copy running-config startup-config`	Saves the configuration to NVRAM.

Austin Router

`Austin(config)#interface serial0/0/0`	Enters interface configuration mode.
`Austin(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Austin(config-if)#ip address 172.16.2.2 255.255.255.0`	Assigns an IP address and netmask.
`Austin(config-if)#ip ospf network broadcast`	Changes the network type from the default nonbroadcast to broadcast.
`Austin(config-if)#ip ospf priority 0`	Sets the priority to 0 for the DR/BDR election process. Austin will not participate in the election process.
`Austin(config-if)#frame-relay map ip 172.16.2.1 150 broadcast`	Maps the remote IP address to local DLCI 150. Broadcast and multicasts will now be forwarded.
`Austin(config-if)#frame-relay map ip 172.16.2.3 150 broadcast`	Maps the remote IP address to local DLCI 150. Broadcast and multicasts will now be forwarded.
`Austin(config-if)#frame-relay map ip 172.16.2.4 150 broadcast`	Maps the remote IP address to local DLCI 150. Broadcast and multicasts will now be forwarded.
`Austin(config-if)#no shutdown`	Enables the interface.
`Austin(config-if)#exit`	Returns to global configuration mode.
`Austin(config)#router ospf 1`	Starts OSPF process 1.
`Austin(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Austin(config-router)#exit`	Returns to global configuration mode.
`Austin(config)#exit`	Returns to privileged mode.
`Austin#copy running-config startup-config`	Saves the configuration to NVRAM.

Galveston Router

`Galveston(config)#interface serial0/0/0`	Enters interface configuration mode.
`Galveston(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Galveston(config-if)#ip address 172.16.2.3 255.255.255.0`	Assigns an IP address and netmask.
`Galveston(config-if)#ip ospf network broadcast`	Changes the network type from the default nonbroadcast to broadcast.
`Galveston(config-if)#ip ospf priority 0`	Sets the priority to 0 for the DR/BDR election process. Galveston will not participate in the election process.
`Galveston(config-if)#frame-relay map ip 172.16.2.1 151 broadcast`	Maps the remote IP address to local DLCI 151. Broadcast and multicasts will now be forwarded.
`Galveston(config-if)#frame-relay map ip 172.16.2.2 151 broadcast`	Maps the remote IP address to local DLCI 151. Broadcast and multicasts will now be forwarded.
`Galveston(config-if)#frame-relay map ip 172.16.2.4 151 broadcast`	Maps the remote IP address to local DLCI 151. Broadcast and multicasts will now be forwarded.
`Galveston(config-if)#no shutdown`	Enables the interface.
`Galveston(config-if)#exit`	Returns to global configuration mode.
`Galveston(config)#router ospf 1`	Starts OSPF process 1.
`Galveston(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x.x* will run OSPF and be placed into area 0."
`Galveston(config-router)#exit`	Returns to global configuration mode.
`Galveston(config)#exit`	Returns to privileged mode.
`Galveston#copy running-config startup-config`	Saves the configuration to NVRAM.

Laredo Router

`Laredo(config)#interface serial0/0/0`	Enters interface configuration mode.
`Laredo(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.

`Laredo(config-if)#ip address 172.16.2.4 255.255.255.0`	Assigns an IP address and netmask.
`Laredo(config-if)#ip ospf network broadcast`	Changes the network type from the default nonbroadcast to broadcast.
`Laredo(config-if)#ip ospf priority 0`	Sets the priority to 0 for the DR/BDR election process. Laredo will not participate in the election process.
`Laredo(config-if)#frame-relay map ip 172.16.2.1 152 broadcast`	Maps the remote IP address to local DLCI 152. Broadcast and multicasts will now be forwarded.
`Laredo(config-if)#frame-relay map ip 172.16.2.2 152 broadcast`	Maps the remote IP address to local DLCI 152. Broadcast and multicasts will now be forwarded.
`Laredo(config-if)#frame-relay map ip 172.16.2.3 152 broadcast`	Maps the remote IP address to local DLCI 152. Broadcast and multicasts will now be forwarded.
`Laredo(config-if)#no shutdown`	Enables the interface.
`Laredo(config-if)#exit`	Returns to global configuration mode.
`Laredo(config)#router ospf 1`	Starts OSPF process 1.
`Laredo(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Laredo(config-router)#exit`	Returns to global configuration mode.
`Laredo(config)#exit`	Returns to privileged mode.
`Laredo#copy running-config startup-config`	Saves the configuration to NVRAM.

Configuration Example: OSPF and Point-to-Multipoint Networks

Figure 3-6 shows the network topology for the configuration that follows, which demonstrates how to configure OSPF on a point-to-multipoint network using the commands covered in this chapter.

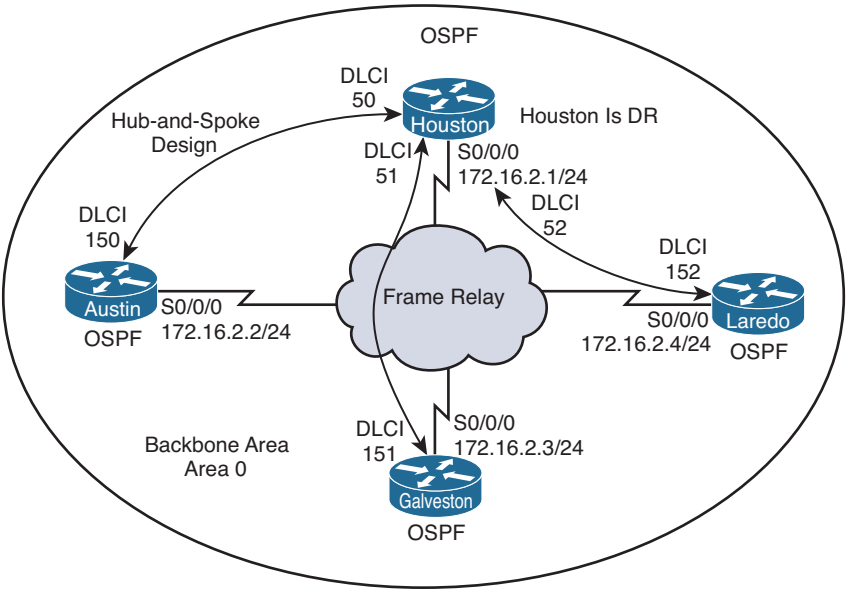


Figure 3-6 Network Topology for OSPF Configuration on a Point-to-Multipoint Network

Houston Router

`Houston(config)#interface serial0/0/0`	Enters interface configuration mode.
`Houston(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Houston(config-if)#ip address 172.16.2.1 255.255.255.0`	Assigns an IP address and netmask.
`Houston(config-if)#ip ospf network point-to-multipoint`	Changes the network type from the default nonbroadcast to point-to-multipoint.
`Houston(config-if)#frame-relay map ip 172.16.2.2 50 broadcast`	Maps the remote IP address to local DLCI 50.
`Houston(config-if)# frame-relay map ip 172.16.2.3 51 broadcast`	Maps the remote IP address to local DLCI 51.
`Houston(config-if)# frame-relay map ip 172.16.2.4 52 broadcast`	Maps the remote IP address to local DLCI 52.
`Houston(config-if)#no shutdown`	Enables the interface.
`Houston(config-if)#exit`	Returns to global configuration mode.
`Houston(config)#router ospf 1`	Enables OSPF process 1.
`Houston(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x.x* will run OSPF and be placed into area 0."

`Houston(config-router)#exit`	Returns to global configuration mode.
`Houston(config)#exit`	Returns to privileged mode.
`Houston#copy running-config startup-config`	Saves the configuration to NVRAM.

Austin Router

`Austin(config)#interface serial0/0/0`	Enters serial interface mode.
`Austin(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Austin(config-if)#ip address 172.16.2.2 255.255.255.0`	Assigns an IP address and netmask.
`Austin(config-if)#ip ospf network point-to-multipoint`	Changes the network type from the default nonbroadcast to point-to-multipoint.
`Austin(config-if)# frame-relay map ip 172.16.2.1 150 broadcast`	Maps the remote IP address to local DLCI 150.
`Austin(config-if)# frame-relay map ip 172.16.2.3 150 broadcast`	Maps the remote IP address to local DLCI 150.
`Austin(config-if)# frame-relay map ip 172.16.2.4 150 broadcast`	Maps the remote IP address to local DLCI 150.
`Austin(config-if)#no shutdown`	Enables the interface.
`Austin(config-if)#exit`	Returns to global configuration mode.
`Austin(config)#router ospf 1`	Starts OSPF process 1.
`Austin(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x.x* will run OSPF and be placed into area 0."
`Austin(config-router)#exit`	Returns to global configuration mode.
`Austin(config)#exit`	Returns to privileged mode.
`Austin#copy running-config startup-config`	Saves the configuration to NVRAM.

Galveston Router

`Galveston(config)#interface serial0/0/0`	Enters interface configuration mode.
`Galveston(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Galveston(config-if)#ip address 172.16.2.3 255.255.255.0`	Assigns an IP address and netmask.
`Galveston(config-if)#ip ospf network point-to-multipoint`	Changes the network type from the default nonbroadcast to point-to-multipoint.

`Galveston(config-if)# frame-relay map ip 172.16.2.1 151 broadcast`	Maps the remote IP address to local DLCI 151.
`Galveston(config-if)# frame-relay map ip 172.16.2.2 151 broadcast`	Maps the remote IP address to local DLCI 151.
`Galveston(config-if)# frame-relay map ip 172.16.2.4 151 broadcast`	Maps the remote IP address to local DLCI 151.
`Galveston(config-if)#no shutdown`	Enables the interface.
`Galveston(config-if)#exit`	Returns to global configuration mode.
`Galveston(config)#router ospf 1`	Starts OSPF process 1.
`Galveston(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x.x* will run OSPF and be placed into area 0."
`Galveston(config-router)#exit`	Returns to global configuration mode.
`Galveston(config)#exit`	Returns to privileged mode.
`Galveston#copy running-config startup-config`	Saves the configuration to NVRAM.

Laredo Router

`Laredo(config)#interface serial0/0/0`	Enters interface configuration mode.
`Laredo(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Laredo(config-if)#ip address 172.16.2.4 255.255.255.0`	Assigns an IP address and netmask.
`Laredo(config-if)#ip ospf network point-to-multipoint`	Changes the network type from the default nonbroadcast to point-to-multipoint.
`Laredo(config-if)# frame-relay map ip 172.16.2.1 152 broadcast`	Maps the remote IP address to local DLCI 152.
`Laredo(config-if)#frame-relay map ip 172.16.2.2 152 broadcast`	Maps the remote IP address to local DLCI 152.
`Laredo(config-if)#frame-relay map ip 172.16.2.3 152 broadcast`	Maps the remote IP address to local DLCI 152.
`Laredo(config-if)#no shutdown`	Enables the interface.
`Laredo(config-if)#exit`	Returns to global configuration mode.
`Laredo(config)#router ospf 1`	Starts OSPF process 1.
`Laredo(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x.x* will run OSPF and be placed into area 0."
`Laredo(config-router)#exit`	Returns to global configuration mode.
`Laredo(config)#exit`	Returns to privileged mode.
`Laredo#copy running-config startup-config`	Saves the configuration to NVRAM.

Configuration Example: OSPF and Point-to-Point Networks Using Subinterfaces

Figure 3-7 shows the network topology for the configuration that follows, which demonstrates how to configure OSPF on a point-to-point network using subinterfaces, using the commands covered in this chapter.

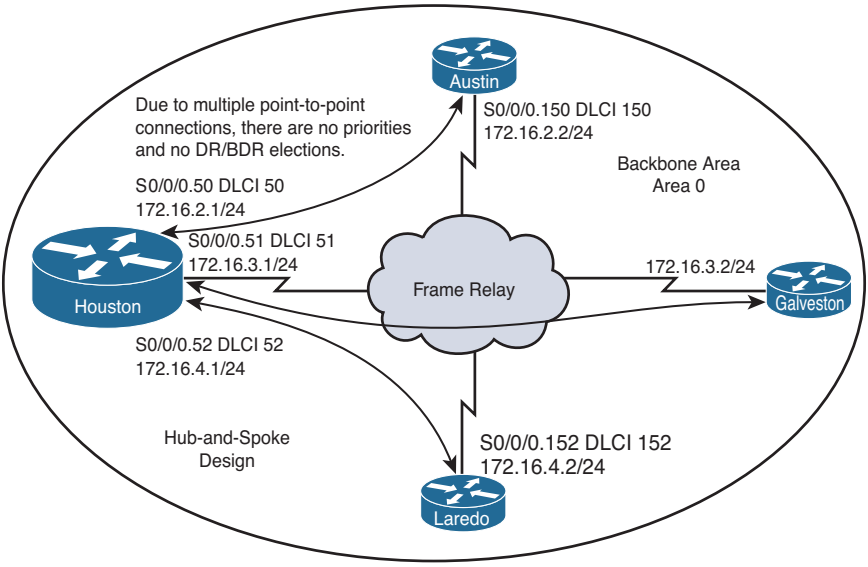


Figure 3-7 Network Topology for OSPF Configuration on a Point-to-Point Network Using Subinterfaces

Houston Router

`Houston(config)#interface serial0/0/0`	Enters interface configuration mode.
`Houston(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Houston(config-if)#no shutdown`	Enables the interface.
`Houston(config-if)#interface serial 0/0/0.50 point-to-point`	Creates a subinterface.
`Houston(config-subif)#description Link to Austin`	Creates a locally significant description of the interface.
`Houston(config-subif)#ip address 172.16.2.1 255.255.255.252`	Assigns an IP address and netmask.
`Houston(config-subif)#frame-relay interface-dlci 50`	Assigns a DLCI to the subinterface.
`Houston(config-subif)#exit`	Returns to interface configuration mode.
`Houston(config-if)#interface serial0/0/0.51 point-to-point`	Creates a subinterface.

`Houston(config-subif)#description Link to Galveston`	Creates a locally significant description of the interface.
`Houston(config-subif)#ip address 172.16.3.1 255.255.255.252`	Assigns an IP address and netmask.
`Houston(config-subif)#frame-relay interface-dlci 51`	Assigns a DLCI to the subinterface.
`Houston(config-subif)#exit`	Returns to interface configuration mode.
`Houston(config-if)#interface serial0/0/0.52 point-to-point`	Creates a subinterface.
`Houston(config-subif)#description Link to Laredo`	Creates a locally significant description of the interface.
`Houston(config-subif)#ip address 172.16.4.1 255.255.255.252`	Assigns an IP address and netmask.
`Houston(config-subif)#frame-relay interface-dlci 52`	Assigns a DLCI to the subinterface.
`Houston(config-subif)#exit`	Returns to interface configuration mode.
`Houston(config-if)#exit`	Returns to global configuration mode.
`Houston(config)#router ospf 1`	Starts OSPF process 1.
`Houston(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Houston(config-router)#exit`	Returns to global configuration mode.
`Houston(config)#exit`	Returns to privileged mode.
`Houston#copy running-config startup-config`	Saves the configuration to NVRAM.

Austin Router

`Austin(config)#interface serial0/0/0`	Enters interface configuration mode.
`Austin(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Austin(config-if)#no shutdown`	Enables the interface.
`Austin(config-if)#interface serial0/0/0.150 point-to-point`	Creates a subinterface.
`Austin(config-subif)#description Link to Houston`	Creates a locally significant description of the interface.
`Austin(config-subif)#ip address 172.16.2.2 255.255.255.252`	Assigns an IP address and netmask.
`Austin(config-subif)#frame-relay interface-dlci 150`	Assigns a DLCI to the subinterface.
`Austin(config-subif)#exit`	Returns to interface configuration mode.
`Austin(config-if)#exit`	Returns to global configuration mode.

`Austin(config)#router ospf 1`	Starts OSPF process 1.
`Austin(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x.x* will run OSPF and be placed into area 0."
`Austin(config-router)#exit`	Returns to global configuration mode.
`Austin(config)#exit`	Returns to privileged mode.
`Austin#copy running-config startup-config`	Saves the configuration to NVRAM.

Galveston Router

`Galveston(config)#interface serial0/0/0`	Enters interface configuration mode.
`Galveston(config-if)# encapsulation frame-relay`	Enables Frame Relay encapsulation.
`Galveston(config-if)#no shutdown`	Enables the interface.
`Galveston(config-if)#interface serial0/0/0.151 point-to-point`	Creates a subinterface.
`Galveston(config-subif)# description Link to Houston`	Creates a locally significant description of the interface.
`Galveston(config-subif)#ip address 172.16.3.2 255.255.255.252`	Assigns an IP address and netmask.
`Galveston(config-subif)#frame-relay interface-dlci 151`	Assigns a DLCI to the subinterface.
`Galveston(config-subif)#exit`	Returns to interface configuration mode.
`Galveston(config-if)#exit`	Returns to global configuration mode.
`Galveston(config)#router ospf 1`	Starts OSPF process 1.
`Galveston(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x.x* will run OSPF and be placed into area 0."
`Galveston(config-router)#exit`	Returns to global configuration mode.
`Galveston(config)#exit`	Returns to privileged mode.
`Galveston#copy running-config startup-config`	Saves the configuration to NVRAM.

Laredo Router

`Laredo(config)#interface serial0/0/0`	Enters interface configuration mode.
`Laredo(config-if)#encapsulation frame-relay`	Enables Frame Relay encapsulation.

`Laredo(config-if)#no shutdown`	Enables the interface.
`Laredo(config-if)#interface serial0/0/0.152 point-to-point`	Creates a subinterface.
`Laredo(config-subif)#description Link to Houston`	Creates a locally significant description of the interface.
`Laredo(config-subif)#ip address 172.16.4.2 255.255.255.252`	Assigns an IP address and netmask.
`Laredo(config-subif)#frame-relay interface-dlci 152`	Assigns a DLCI to the subinterface.
`Laredo(config-subif)#exit`	Returns to interface configuration mode.
`Laredo(config-if)#exit`	Returns to global configuration mode.
`Laredo(config)#router ospf 1`	Starts OSPF process 1.
`Laredo(config-router)#network 172.16.0.0 0.0.255.255 area 0`	Read this line to say, "Any interface with an IP address of 172.16.*x*.*x* will run OSPF and be placed into area 0."
`Laredo(config-router)#exit`	Returns to global configuration mode.
`Laredo(config)#exit`	Returns to privileged mode.
`Laredo#copy running-config startup-config`	Saves the configuration to NVRAM.

Configuration Example: IPv6 and OSPFv3

Figure 3-8 shows the network topology for the configuration that follows, which demonstrates how to configure IPv6 and OSPFv3 using the commands covered in this chapter.

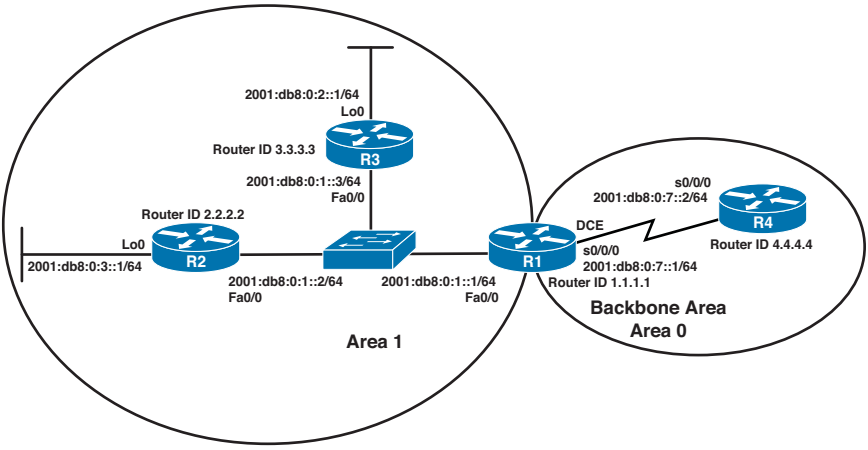


Figure 3-8 Network Topology for IPv6 and OSPFv3 Configuration

R3 Router

`R3(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router. This command is required before any IPv6 routing protocol can be configured.
`R3(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`R3(config-if)#ipv6 address 2001:db8:0:1::3/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
`R3(config-if)#ipv6 ospf 1 area 1`	Enables OSPFv3 on the interface and places this interface into area 1.
`R3(config-if)#no shutdown`	Enables the interface.
`R3(config-if)#interface loopback0`	Moves to interface configuration mode.
`R3(config-if)#ipv6 address 2001:db8:0:2::1/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
`R3(config-if)#ipv6 ospf 1 area 1`	Enables OSPFv3 on the interface and places this interface into area 1.
`R3(config-if)#exit`	Moves to global configuration mode.
`R3(config)#ipv6 router ospf 1`	Moves to OSPFv3 router config mode
`R3(config-rtr)#router-id 3.3.3.3`	Sets a manually configured router ID
`R3(config-rtr)#exit`	Returns to global configuration mode.
`R3(config)#exit`	Moves to privileged mode.
`R3#copy running-config startup-config`	Saves the configuration to NVRAM.

R2 Router

`R2(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router. This command is required before any IPv6 routing protocol can be configured.
`R2(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`R2(config-if)#ipv6 address 2001:db8:0:1::2/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
`R2(config-if)#ipv6 ospf 1 area 1`	Enables OSPFv3 on the interface and places this interface into area 1.
`R2(config-if)#no shutdown`	Enables the interface.
`R2(config-if)#interface loopback0`	Moves to interface configuration mode.

R2(config-if)#**ipv6 address 2001:db8:0:3::1/64**	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
R2(config-if)#**ipv6 ospf 1 area 1**	Enables OSPFv3 on the interface and places this interface into area 1.
R2(config-if)#**no shutdown**	Enables the interface.
R2(config-if)#**exit**	Moves to global configuration mode.
R2(config)#**ipv6 router ospf 1**	Moves to OSPFv3 router config mode
R2(config-rtr)#**router-id 2.2.2.2**	Sets a manually configured router ID
R2(config-rtr)#**exit**	Returns to global configuration mode
R2(config)#**exit**	Moves to privileged mode.
R2#**copy running-config startup-config**	Saves the configuration to NVRAM.

R1 Router

R1(config)#**ipv6 unicast-routing**	Enables the forwarding of IPv6 unicast datagrams globally on the router. This command is required before any IPv6 routing protocol can be configured.
R1(config)#**interface fastethernet0/0**	Moves to interface configuration mode.
R1(config-if)#**ipv6 address 2001:db8:0:1::1/64**	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
R1(config-if)#**ipv6 ospf 1 area 1**	Enables OSPFv3 on the interface and places this interface into area 1.
R1(config-if)#**no shutdown**	Enables the interface.
R1(config-if)#**interface serial0/0/0**	Moves to interface configuration mode.
R1(config-if)#**ipv6 address 2001:db8:0:7::1/64**	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
R1(config-if)#**ipv6 ospf 1 area 0**	Enables OSPFv3 on the interface and places this interface into area 0.
R1(config-if)#**clock rate 56000**	Assigns a clock rate to this interface.
R1(config-if)#**no shutdown**	Enables the interface.
R1(config-if)#**exit**	Moves to global configuration mode.
R1(config)#**ipv6 router ospf 1**	Moves to OSPFv3 router config mode.
R1(config-rtr)#**router-id 1.1.1.1**	Sets a manually configured router ID.
R1(config-rtr)#**exit**	Returns to global configuration mode.
R1(config)#**exit**	Moves to privileged mode.
R1#**copy running-config startup-config**	Saves the configuration to NVRAM.

R4 Router

`R4(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router. This command is required before any IPv6 routing protocol can be configured.
`R4(config)#interface serial0/0/0`	Moves to interface configuration mode.
`R4(config-if)#ipv6 address 2001:db8:0:7::2/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
`R4(config-if)#ipv6 ospf 1 area 0`	Enables OSPFv3 on the interface and places this interface into area 1.
`R4(config-if)#no shutdown`	Enables the interface.
`R4(config-if)#exit`	Moves to global configuration mode.
`R4(config)#ipv6 router ospf 1`	Moves to OSPFv3 router config mode.
`R4(config-rtr)#router-id 4.4.4.4`	Sets a manually configured router ID.
`R4(config-rtr)#exit`	Returns to global configuration mode.
`R4(config)#exit`	Moves to privileged mode.
`R4#copy running-config startup-config`	Saves the configuration to NVRAM.

Configuration Example: OSPFv3 with Address Families

Figure 3-9 shows the network topology for the configuration that follows, which demonstrates how to configure OSPFv3 address families using the commands covered in this chapter.

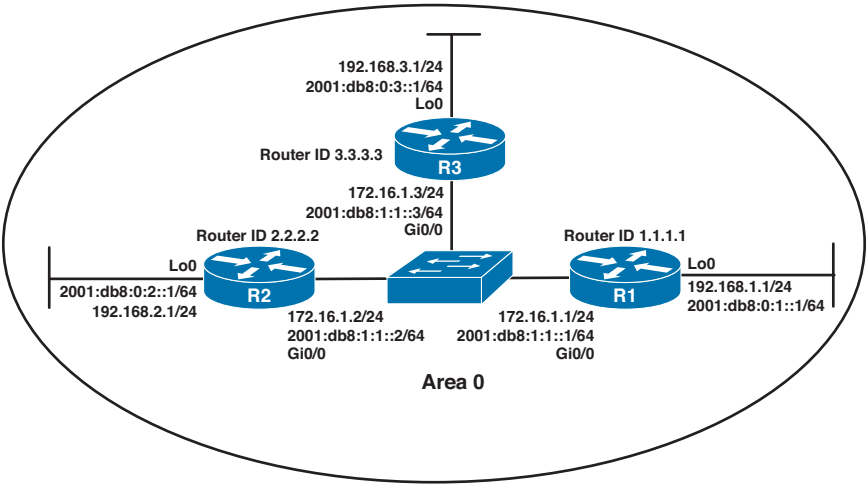


Figure 3-9 Network Topology for IPv6 and OSPFv3 Configuration

R1 Router

`R1(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router. This command is required before any IPv6 routing protocol can be configured.
`R1(config)#interface loopback0`	Moves to interface configuration mode.
`R1(config-if)#ip address 192.168.1.1 255.255.255.0`	Assigns an IP address and netmask.
`R1(config-if)#ipv6 address 2001:DB8:0:1::1/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
`R1(config-if)#interface gigabitethernet0/0`	Moves to interface configuration mode.
`R1(config-if)#ip address 172.16.1.1 255.255.255.0`	Assigns an IP address and netmask.
`R1(config-if)#ipv6 address 2001:DB8:1:1::1/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
`R1(config-if)#no shutdown`	Enables the interface.
`R1(config-if)#exit`	Returns to global configuration mode.
`R1(config)#router ospfv3 1`	Enables OSPFv3 router configuration mode for the IPv4 or IPv6 address family.
`R1(config-router)#log-adjacency-changes`	Configures the router to send a syslog message when an OSPFv3 neighbor goes up or down.
`R1(config-router)#router-id 1.1.1.1`	Configures a fixed router ID.
`R1(config-router)#address family ipv6 unicast`	Enters IPv6 address family configuration mode for OSPFv3.
`R1(config-router-af)#passive-interface Loopback 0`	Prevents interface loopback 0 from exchanging any OSPF packets, including Hello packets.
`R1(config-router-af)#address family ipv4 unicast`	Enters IPv4 address family configuration mode for OSPFv3.
`R1(config-router-af)#passive-interface Loopback 0`	Prevents interface loopback 0 from exchanging any OSPF packets, including Hello packets.
`R1(config-router-af)#exit`	Returns to OSPFv3 router configuration mode.
`R1(config-router)#exit`	Returns to global configuration mode.
`R1(config)#interface loopback0`	Moves to interface configuration mode.
`R1(config-if)#ospfv3 1 ipv6 area 0`	Enables OSPFv3 instance 1 with the IPv6 address family in area 0.

R1(config-if)#**ospfv3 1 ipv4 area 0**	Enables OSPFv3 instance 1 with the IPv4 address family in area 0.
R1(config-if)#**interface gigabitethernet 0/0**	Moves to interface configuration mode.
R1(config-if)#**ospfv3 1 ipv6 area 0**	Enables OSPFv3 instance 1 with the IPv6 address family in area 0.
R1(config-if)#**ospfv3 1 ipv4 area 0**	Enables OSPFv3 instance 1 with the IPv4 address family in area 0.
R1(config-if)#**exit**	Returns to global configuration mode.
R1(config)#**exit**	Returns to privileged mode.
R1#**copy running-config startup-config**	Copies the running configuration to NVRAM.

R2 Router

R2(config)#**ipv6 unicast-routing**	Enables the forwarding of IPv6 unicast datagrams globally on the router. This command is required before any IPv6 routing protocol can be configured.
R2(config)#**interface loopback0**	Moves to interface configuration mode.
R2(config-if)#**ip address 192.168.2.1 255.255.255.0**	Assigns an IP address and netmask.
R2(config-if)#**ipv6 address 2001:DB8:0:2::1/64**	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
R2(config-if)#**interface gigabitethernet0/0**	Moves to interface configuration mode.
R2(config-if)#**ip address 172.16.1.2 255.255.255.0**	Assigns an IP address and netmask.
R2(config-if)#**ipv6 address 2001:DB8:1:1::2/64**	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
R2(config-if)#**no shutdown**	Enables the interface.
R2(config-if)#**exit**	Returns to global configuration mode.
R2(config)#**router ospfv3 1**	Enables OSPFv3 router configuration mode for the IPv4 or IPv6 address family.
R2(config-router)#**log-adjacency-changes**	Configures the router to send a syslog message when an OSPFv3 neighbor goes up or down.
R2(config-router)#**router-id 2.2.2.2**	Configures a fixed router ID.
R2(config-router)#**address family ipv6 unicast**	Enters IPv6 address family configuration mode for OSPFv3.

R2(config-router-af)#**passive-interface Loopback 0**	Prevents interface loopback 0 from exchanging any OSPF packets, including Hello packets.
R2(config-router-af)#**address family ipv4 unicast**	Enters IPv4 address family configuration mode for OSPFv3.
R2(config-router-af)#**passive-interface Loopback 0**	Prevents interface loopback 0 from exchanging any OSPF packets, including Hello packets.
R2(config-router-af)#**exit**	Returns to OSPFv3 router configuration mode.
R2(config-router)#**exit**	Returns to global configuration mode.
R2(config)#**interface loopback 0**	Moves to interface configuration mode.
R2(config-if)#**ospfv3 1 ipv6 area 0**	Enables OSPFv3 instance 1 with the IPv6 address family in area 0.
R2(config-if)#**ospfv3 1 ipv4 area 0**	Enables OSPFv3 instance 1 with the IPv4 address family in area 0.
R2(config-if)#**interface gigabitethernet 0/0**	Moves to interface configuration mode.
R2(config-if)#**ospfv3 1 ipv6 area 0**	Enables OSPFv3 instance 1 with the IPv6 address family in area 0.
R2(config-if)#**ospfv3 1 ipv4 area 0**	Enables OSPFv3 instance 1 with the IPv4 address family in area 0.
R2(config-if)#**exit**	Returns to global configuration mode.
R2(config)#**exit**	Returns to privileged mode.
R2#**copy running-config startup-config**	Copies the running configuration to NVRAM.

R3 Router

R3(config)#**ipv6 unicast-routing**	Enables the forwarding of IPv6 unicast datagrams globally on the router. This command is required before any IPv6 routing protocol can be configured.
R3(config)#**interface loopback0**	Moves to interface configuration mode.
R3(config-if)#**ip address 192.168.3.1 255.255.255.0**	Assigns an IP address and netmask.
R3(config-if)#**ipv6 address 2001:DB8:0:3::1/64**	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
R3(config-if)#**interface gigabitethernet0/0**	Moves to interface configuration mode.
R3(config-if)#**ip address 172.16.1.3 255.255.255.0**	Assigns an IP address and netmask.

`R3(config-if)#ipv6 address 2001:DB8:1:1::3/64`	Configures a global IPv6 address on the interface and enables IPv6 processing on the interface.
`R3(config-if)#no shutdown`	Enables the interface.
`R3(config-if)#exit`	Returns to global configuration mode.
`R3(config)#router ospfv3 1`	Enables OSPFv3 router configuration mode for the IPv4 or IPv6 address family.
`R3(config-router)#log-adjacency-changes`	Configures the router to send a syslog message when an OSPFv3 neighbor goes up or down.
`R3(config-router)#router-id 3.3.3.3`	Configures a fixed router ID.
`R3(config-router)#address family ipv6 unicast`	Enters IPv6 address family configuration mode for OSPFv3.
`R3(config-router-af)#passive-interface Loopback 0`	Prevents interface loopback 0 from exchanging any OSPF packets, including Hello packets.
`R3(config-router-af)#address family ipv4 unicast`	Enters IPv4 address family configuration mode for OSPFv3.
`R3(config-router-af)#passive-interface Loopback 0`	Prevents interface loopback 0 from exchanging any OSPF packets, including Hello packets.
`R3(config-router-af)#exit`	Returns to OSPFv3 router configuration mode.
`R3(config-router)#exit`	Returns to global configuration mode.
`R3(config)#interface loopback0`	Moves to interface configuration mode.
`R3(config-if)#ospfv3 1 ipv6 area 0`	Enables OSPFv3 instance 1 with the IPv6 address family in area 0.
`R3(config-if)#ospfv3 1 ipv4 area 0`	Enables OSPFv3 instance 1 with the IPv4 address family in area 0.
`R3(config-if)#interface gigabitethernet 0/0`	Moves to interface configuration mode.
`R3(config-if)#ospfv3 1 ipv6 area 0`	Enables OSPFv3 instance 1 with the IPv6 address family in area 0.
`R3(config-if)#ospfv3 1 ipv4 area 0`	Enables OSPFv3 instance 1 with the IPv4 address family in area 0.
`R3(config-if)#exit`	Returns to global configuration mode.
`R3(config)#exit`	Returns to privileged mode.
`R3#copy running-config startup-config`	Copies the running configuration to NVRAM.

CHAPTER 4

Configuration of Redistribution

This chapter provides information about the following redistribution topics:

- Defining seed and default metrics
- Redistributing connected networks
- Redistributing static routes
- Redistributing subnets into OSPF
- Assigning E1 or E2 routes in OSPF
- Redistributing OSPF internal and external routes
- Configuration example: route redistribution for IPv4
- Configuration example: route redistribution for IPv6
- Verifying route redistribution
- Route filtering using the **distribute-list** command
    - Configuration example: inbound and outbound distribute list route filters
    - Configuration example: controlling redistribution with outbound distribute lists
    - Verifying route filters
- Route filtering using prefix lists
    - Configuration example: using a distribute list that references a prefix list to control redistribution
    - Verifying prefix lists
- Using route maps with route redistribution
    - Configuration example: route maps
- Manipulating redistribution using route tagging
- Changing administrative distance for internal and external routes
- Passive interfaces

Defining Seed and Default Metrics

`Router(config)#router eigrp 100`	Starts the EIGRP routing process.
`Router(config-router)#network 172.16.0.0`	Specifies which network to advertise in EIGRP.

`Router(config-router)#` **`redistribute rip`** `Router(config-router)#`**`default-metric 1000 100 250 1 1500`** Or `Router(config-router)#` **`redistribute rip metric 1000 100 250 1 1500`**	Redistributes routes learned from RIP into EIGRP. The metrics assigned to these learned routes will be calculated using the following components: 1000 = Bandwidth in Kbps 100 = Delay in tens of microseconds 255 = Reliability out of 255 1 = Load out of 255 1500 = Maximum transmission unit (MTU) size The **metric** keyword in the second option assigns a starting EIGRP metric that is calculated using the following components: 1000, 100, 255, 1 1500.

NOTE: The values used in this command constitute the seed metric for these RIP routes being redistributed into EIGRP. The seed metric is the initial value of an imported route and it must be consistent with the destination protocol.

NOTE: The default seed metrics are as follows:

- Connected: 1
- Static: 1
- RIP: Infinity
- EIGRP: Infinity
- OSPF: 20 for all except for BGP, which is 1
- BGP: BGP metric is set to IGP metric value

NOTE: If both the **metric** keyword in the **redistribute** command and the **default-metric** command are used, the value of the **metric** keyword in the **redistribute** command takes precedence.

TIP: If a value is not specified for the **metric** option, and no value is specified using the **default-metric** command, the default metric value is 0, except for Open Shortest Path First (OSPF) Protocol, where the default cost is 20. Routing Information Protocol (RIP) and Enhanced Interior Gateway Routing Protocol (EIGRP) must have the appropriate metrics assigned to any redistributed routes; otherwise, redistribution will not work. Border Gateway Protocol (BGP) will use the Internal Gateway Protocol (IGP) metric, while both connected networks and static routes will receive an initial default value of 1.

TIP: The **default-metric** command is useful when routes are being redistributed from more than one source because it eliminates the need for defining the metrics separately for each redistribution.

TIP: Redistributed routes between EIGRP processes do not need metrics configured. Redistributed routes are tagged as EIGRP external routes and will appear in the routing table with a code of D EX.

Redistributing Connected Networks

`Router(config)#router ospf 1`	Starts the OSPF routing process.
`Router(config-router)#` `redistribute connected`	Redistributes all directly connected networks.
	NOTE: It is not necessary to redistribute networks that are already configured under the routing protocol.
	NOTE: The **connected** keyword refers to routes that are established automatically by virtue of having enabled IP on an interface. For routing protocols such as OSPF, Intermediate System-to-Intermediate System (IS-IS), and EIGRP, these routes are redistributed as external to the autonomous system.

`Router(config-router)#`
`redistribute connected`
`metric 50`
	Redistributes all directly connected networks and assigns them a starting metric of 50.

	NOTE: The **redistribute connected** command is *not* affected by the **default-metric** command.

Redistributing Static Routes

`Router(config)#ip route 10.1.1.0` `255.255.255.0 serial 0/0/0`	Creates a static route for network 10.1.1.0/24 exiting out of interface Serial 0/0/0
`Router(config)#router eigrp 10`	Starts the EIGRP routing process
`Router(config-router)#` `redistribute static`	Redistributes static routes on this router into the EIGRP routing process

Redistributing Subnets into OSPF

`Router(config)#router ospf 1`	Starts the OSPF routing process.

`Router(config-router)#`
`redistribute eigrp 10`
`metric 100 subnets`
	Redistributes routes learned from EIGRP autonomous system 10. A metric of 100 is assigned to all routes. Subnets will also be redistributed.

	NOTE: Without the **subnets** keyword, no subnets will be redistributed into the OSPF domain. (Only routes that are in the routing table with the default classful mask will be redistributed.)

Assigning E1 or E2 Routes in OSPF

`Router(config)#router ospf 1`	Starts the OSPF routing process.
`Router(config-router)#` **`redistribute eigrp 1 metric-type 1`**	Redistributes routes learned from EIGRP autonomous system 1. Routes will be advertised as E1 routes.
	NOTE: If the **metric-type** argument is not used, routes will be advertised by default in OSPF as E2 routes. E2 routes have a default fixed cost of 20 associated with them, but this value can be changed with the **metric** keyword. The metric will not change as the route is propagated throughout the OSPF area. E1 routes will have internal area costs added to the seed metric.

TIP: Use external type 1 (E1) routes when there are multiple Autonomous System Border Routers (ASBRs) advertising an external route to the same autonomous system to avoid suboptimal routing (see Figure 4-1).

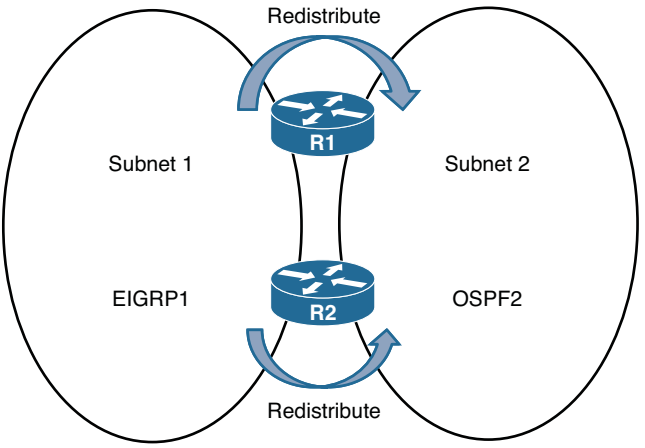


Figure 4-1 Network Topology with Two ASBRs

TIP: Use external type 2 (E2) routes if only one ASBR is advertising an external route to the AS (see Figure 4-2).

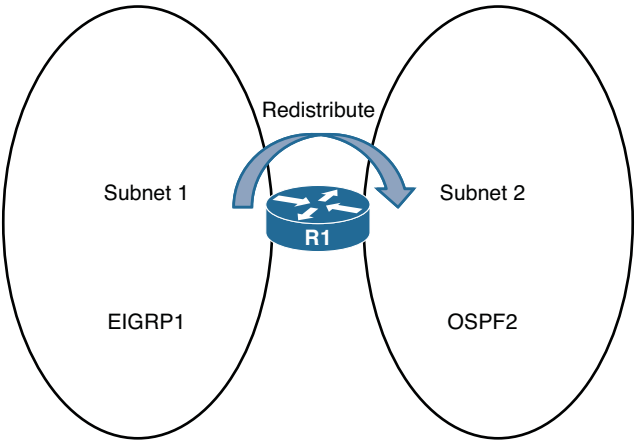


Figure 4-2 Network Topology with One ASBR

Redistributing OSPF Internal and External Routes

`Router(config)#router eigrp 10`	Starts the EIGRP routing process for autonomous system 10.
`Router(config-router)#` **`redistribute ospf 1 match internal external 1 external 2`**	Redistributes routes learned from OSPF process ID 1. The keywords **match internal external 1** and **external 2** instruct EIGRP to only redistribute internal, external type 1 and type 2 OSPF routes.
	NOTE: The default behavior when redistributing OSPF routes is to redistribute all routes—internal, external 1, and external 2. The keywords **match internal external 1** and **external 2** are required only if router behavior is to be modified.

Configuration Example: Route Redistribution for IPv4

Figure 4-3 shows the network topology for the configuration that follows, which demonstrates how to configure single point two-way basic redistribution between EIGRP and OSPF for IPv4, using the commands covered in this chapter. For this configuration example, assume that EIGRP and OSPF routing has been configured correctly on all four routers.

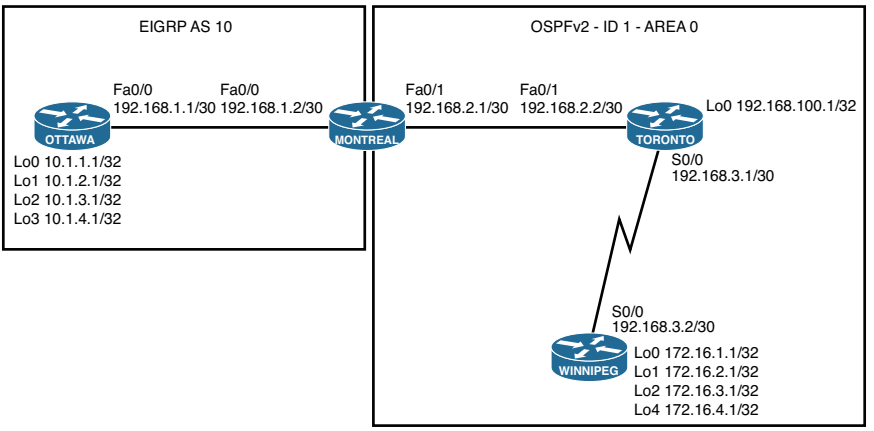


Figure 4-3 Network Topology for IPv4 Route Redistribution

`MONTREAL(config)#router eigrp 10`	Enters EIGRP configuration mode.
`MONTREAL(config-router)#` **`redistribute ospf 1 metric 1500 10 255 1 1500`**	Redistributes routes from OSPF process ID 1 into EIGRP AS 10 and assigns a seed metric to these routes.
`MONTREAL(config-router)#exit`	Returns to global configuration mode.
`MONTREAL(config)#router ospf 1`	Enters OSPF configuration mode.
`MONTREAL(config-router)#` **`redistribute eigrp 10 subnets`**	Redistributes classless routes from EIGRP autonomous system 10 into OSPF process ID 1 as external type 2 (E2) with a metric of 20, which is fixed and does not change across the OSPF domain.
	NOTE: Omitting the **subnets** keyword is a common configuration error. Without this keyword, only networks in the routing table with a classful mask will be redistributed. Subnets will not be redistributed, and subnets will not be automatically summarized and redistributed.
`MONTREAL(config-router)#` **`redistribute eigrp 10 metric-type 1 subnets`**	Redistributes classless routes from EIGRP autonomous system 10 into OSPF process ID 1 as external type 1 (E1). Type 1 external routes calculate the cost by adding the external cost (20) to the internal cost of each link that the packet crosses.

Configuration Example: Route Redistribution for IPv6

Figure 4-4 shows the network topology for the configuration that follows, which demonstrates how to configure single point two-way basic redistribution between EIGRP and OSPF for IPv6, using the commands covered in this chapter. For this configuration example, assume that EIGRP and OSPF routing for IPv6 has been configured correctly on all four routers.

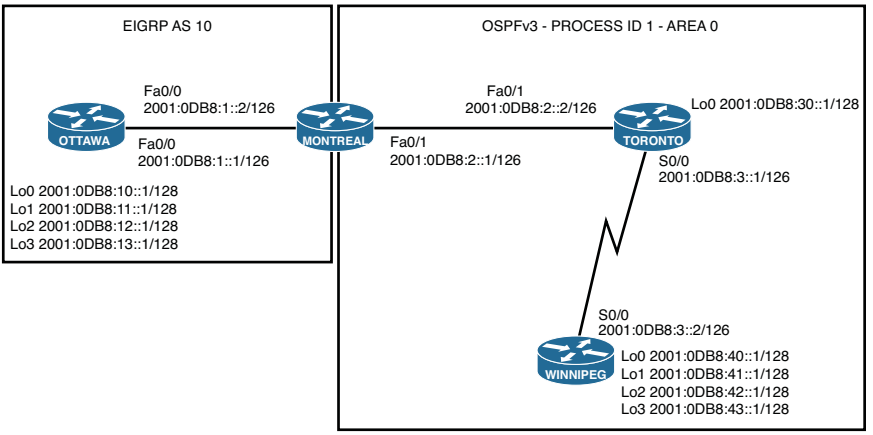


Figure 4-4 Network Topology for IPv6 Route Redistribution

`MONTREAL(config)#ipv6 router eigrp 10`	Enters IPv6 EIGRP configuration mode.
`MONTREAL(config-router)#redistribute ospf 1 metric 1500 10 255 1 1500 include-connected`	Redistributes IPv6 routes from OSPF process ID 1 into EIGRP autonomous system 10 and assigns a seed metric to these routes.
	NOTE: With the **include-connected** command, you instruct the target routing protocol to redistribute the routes that are learned by the source protocol and also the connected interfaces if the source routing protocol is running on them.
`MONTREAL(config-router)#exit`	Returns to global configuration mode.
`MONTREAL(config)#ipv6 router ospf 1`	Enters IPv6 OSPF configuration mode.
`MONTREAL(config-router)#redistribute eigrp 10 include-connected`	Redistributes IPv6 routes from EIGRP autonomous system 10 into OSPF process ID 1 as external type 2 (E2) with a metric of 20, which is fixed and does not change across the OSPF domain.

`MONTREAL(config-router)#` **`redistribute eigrp 10 metric-type 1 include-connected`**	Redistributes IPv6 routes from EIGRP autonomous system 10 into OSPF process ID 1 as external type 1 (E1). Type 1 external routes calculate the cost by adding the external cost (20) to the internal cost of each link that the packet crosses.
	NOTE: The **subnets** keyword does not exist in OSPFv3 redistribution configuration.

Verifying Route Redistribution

`Router#`**`show ip route`** `Router#`**`show ipv6 route`**	Displays the current state of the routing table
`Router#`**`show ip eigrp topology`** `Router#`**`show ipv6 eigrp topology`**	Displays the EIGRP topology table
`Router#`**`show ip protocols`** `Router#`**`show ipv6 protocols`**	Displays parameters and the current state of any active routing process
`Router#`**`show ip rip database`** `Router#`**`show ipv6 rip database`**	Displays summary address entries in the RIP routing database
`Router#`**`show ip ospf database`** `Router#`**`show ipv6 ospf database`**	Displays the link-state advertisement (LSA) types within the link-state database (LSDB)

Route Filtering Using the distribute-list Command

`Router(config)#`**`router eigrp 10`**	Starts the EIGRP routing process for autonomous system 10
`Router(config-router)#`**`distribute-list 1 in`**	Creates an incoming global distribute list that refers to access control list (ACL) 1
`Router(config-router)#`**`distribute-list 2 out`**	Creates an outgoing global distribute list that refers to ACL 2
`Router(config-router)#`**`distribute-list 3 in fastethernet0/0`**	Creates an incoming distribute list for interface FastEthernet0/0 and refers to ACL 3
`Router(config-router)#`**`distribute-list 4 out serial0/0/0`**	Creates an outgoing distribute list for interface Serial0/0/0 and refers to ACL 4
`Router(config-router)#`**`distribute-list 5 out ospf 1`**	Filters updates advertised from OSPF process ID 1 into EIGRP autonomous system 10 according to ACL 5

Configuration Example: Inbound and Outbound Distribute List Route Filters

Figure 4-5 shows the network topology for the configuration that follows, which demonstrates how to configure inbound and outbound route filters to control routing updates using the commands covered in this chapter. Assume that all basic configurations and EIGRP routing have been configured correctly.

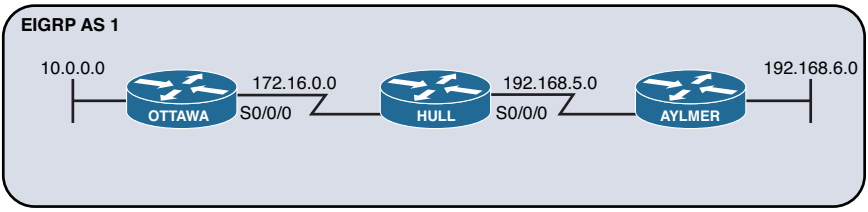


Figure 4-5 Network Topology for Inbound and Outbound Distribute List Route Filters

The first objective is to prevent router AYLMER from learning the 10.0.0.0/8 network using an outbound distribute list on router HULL.

`HULL(config)#access-list 10 deny 10.0.0.0 0.255.255.255`	Creates a standard ACL number 10 and explicitly denies the 10.0.0.0/8 network
`HULL(config)#access-list 10 permit any`	Adds a second line to ACL 10 which permits all other networks
`HULL(config)#router eigrp 1`	Enters EIGRP autonomous system 1 routing process

`HULL(config-router)#distribute-list 10 out`
Or
`HULL(config-router)#distribute-list 10 out serial0/0/0`
	Creates an outbound global distribute list that refers to ACL 10
	Creates an outgoing distribute list for interface Serial0/0/0 that refers to ACL 10

The second objective is to prevent router OTTAWA from learning the 192.168.6.0/24 network using an inbound distribute list on router OTTAWA.

`OTTAWA(config)#access-list 20 deny 192.168.6.0 0.0.0.255`	Creates a standard ACL number 20 and explicitly denies the 192.168.6.0/24 network
`OTTAWA(config)#access-list 20 permit any`	Adds a second line to ACL 20 which permits all other networks
`OTTAWA (config)#router eigrp 1`	Enters EIGRP autonomous system 1 routing process

`OTTAWA(config-router)# distribute-list 20 in`
Or
`OTTAWA(config-router)# distribute-list 20 in serial0/0/0`
	Creates an inbound global distribute list that refers to ACL 20
	Creates an inbound distribute list for interface Serial0/0/0 that refers to ACL 20

Configuration Example: Controlling Redistribution with Outbound Distribute Lists

Figure 4-6 shows the network topology for the configuration that follows, which demonstrates how to control redistribution with an outbound distribute list using the commands covered in this chapter. Assume that all basic configurations and EIGRP and OSPF routing have been configured correctly.

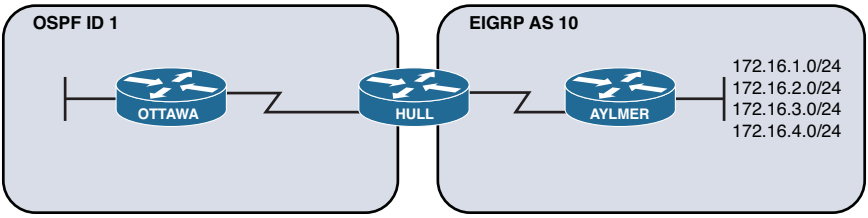


Figure 4-6 Network Topology for Controlling Redistribution with Outbound Distribute Lists

The objective is to prevent networks 172.16.3.0/24 and 172.16.4.0/24 from being redistributed into the OSPF domain.

`HULL(config)#access-list 30 permit 172.16.1.0 0.0.0.255`	Creates a standard ACL number 30 and explicitly permits the 172.16.1.0/24 network.
`HULL (config)#access-list 30 permit 172.16.2.0 0.0.0.255`	Adds a second line to ACL 30 that explicitly permits the 172.16.2.0/24 network.
`HULL(config)#router ospf 1`	Enters OSPF process ID 1 routing process.
`HULL(config-router)#redistribute eigrp 10 subnets`	Redistributes all EIGRP networks into OSPF.
`HULL(config-router)#distribute-list 30 out eigrp 10`	Creates an outbound distribute list to filter routes being redistributed from EIGRP into OSPF.
	NOTE: The implicit "deny any" statement at the end of the access list prevents routing updates about any other network from being advertised. As a result, networks 172.16.3.0/24 and 172.16.4.0/24 will not be redistributed into OSPF.

Verifying Route Filters

`Router#show ip protocols`	Displays the parameters and current state of active routing protocols

```
Routing Protocol is "eigrp 10"
  Outgoing update filter list for all interfaces is 2
   Redistributed ospf 1 filtered by 5
    Serial 0/0/0 filtered by 4
  Incoming update filter list for all interfaces is 1
    FastEthernet0/0 filtered by 3
```

NOTE: For each interface and routing process, Cisco IOS permits the following:

- One incoming global distribute list
- One outgoing global distribute list
- One incoming interface distribute list
- One outgoing interface distribute list
- One outgoing redistribution distribute list

CAUTION: Route filters have *no* effect on LSAs or the LSDB. A basic requirement of link-state routing protocols is that routers in an area must have identical LSDBs.

NOTE: OSPF routes *cannot* be filtered from entering the OSPF database. The **distribute-list in** command filters routes only from entering the routing table, but it doesn't prevent link-state packets (LSP) from being propagated.

The command **distribute-list out** works only on the routes being redistributed by the ASBR into OSPF. It can be applied to external type 2 and external type 1 routes but *not* to intra-area and interarea routes.

Route Filtering Using Prefix Lists

The general syntax for configuring a prefix list is as follows:

```
Router(config)#ip prefix-list list-name [seq seq-value] deny | permit
network/len [ge ge-value] [le le-value]
```

The table that follows describes the parameters for this command.

Parameter	Description
list-name	The name of the prefix list
seq	(Optional) Applies a sequence number to the entry being created or deleted
seq-value	(Optional) Specifies the sequence number
deny	Denies access to matching conditions
permit	Permits access for matching conditions
network/len	(Mandatory) The network number and length (in bits) of the netmask
ge	(Optional) Applies *ge-value* to the range specified

Parameter	Description
ge-value	(Optional) Specifies the lesser value of a range (the "from" portion of the range description)
le	(Optional) Applies *le-value* to the range specified
le-value	(Optional) Specifies the greater value of a range (the "to" portion of the range description)

TIP: You must define a prefix list before you can apply it as a route filter.

TIP: There is an implicit deny statement at the end of each prefix list.

TIP: The range of sequence numbers that can be entered is from 1 to 4,294,967,294. If a sequence number is not entered when configuring this command, a default sequence numbering is applied to the prefix list. The number 5 is applied to the first prefix entry, and subsequent unnumbered entries are incremented by 5.

A router tests for prefix list matches from the lowest sequence number to the highest.

By numbering your **prefix-list** statements, you can add new entries at any point in the list.

The following examples show how you can use the **prefix-list** command to filter networks using some of the more commonly used options.

`Router(config)#ip prefix-list ROSE permit 192.0.0.0/8 le 24`	Creates a prefix list that will accept a netmask of up to 24 bits (**le** meaning less than or equal to) in routes with the prefix 192.0.0.0/8. Because no sequence number is identified, the default number of 5 is applied.
`Router(config)#ip prefix-list ROSE deny 192.0.0.0/8 ge 25`	Creates a prefix list that will deny routes with a netmask of 25 bits or greater (**ge** meaning greater than or equal to) in routes with the prefix 192.0.0.0/8. Because no sequence number is identified, the number 10 is applied—an increment of 5 over the previous statement.
	NOTE: This configuration will permit routes such as 192.2.0.0/16 or 192.2.20.0/24, but will deny a more specific subnet such as 192.168.10.128/25.
`Router(config)#ip prefix-list TOWER permit 10.0.0.0/8 ge 16 le 24`	Creates a prefix list that permits all prefixes in the 10.0.0.0/8 address space that have a netmask of between 16 and 24 bits (greater than or equal to 16 bits, and less than or equal to 24 bits).

`Router(config)#ip prefix-list TEST seq 5 permit 0.0.0.0/0`	Creates a prefix list and assigns a sequence number of 5 to a statement which permits only the default route 0.0.0.0/0.
`Router(config)#ip prefix-list TEST seq 10 permit 0.0.0.0/0 ge 30 le 30`	Creates a prefix list and assigns a sequence number of 10 to a statement that permits any prefix with a netmask of exactly 30 bits.
`Router(config)#ip prefix-list TEST seq 15 permit 0.0.0.0/0 le 32`	Creates a prefix list and assigns a sequence number of 15 to a statement that permits any address or subnet (permit any).
`Router(config)#no ip prefix-list TEST seq 10`	Removes sequence number 10 from the prefix list.

Configuration Example: Using a Distribute List That References a Prefix List to Control Redistribution

Figure 4-7 shows the network topology for the configuration that follows, which demonstrates how to control redistribution with a prefix list using the commands covered in this chapter. Assume that all basic configurations and EIGRP and OSPF routing have been configured correctly.

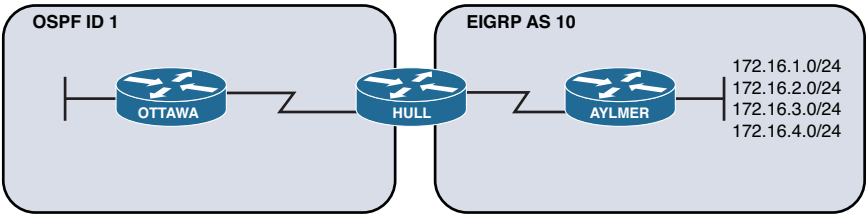


Figure 4-7 Network Topology for Distribute List Configuration with Prefix Lists

The objective is to prevent networks 172.16.3.0/24 and 172.16.4.0/24 from being redistributed into the OSPF domain.

`HULL(config)#ip prefix-list FILTER seq 5 permit 172.16.1.0/24`	Creates a prefix list called FILTER with a first sequence number of 5 that explicitly permits the 172.16.1.0/24 network.
`HULL (config)#ip prefix-list FILTER seq 10 permit 172.16.2.0/24`	Adds a second line to the FILTER prefix list that explicitly permits the 172.16.2.0/24 network.
`HULL(config)#router ospf 1`	Enters OSPF process ID 1 routing process.
`HULL(config-router)# redistribute eigrp 10 subnets`	Redistributes all EIGRP networks into OSPF.

`HULL(config-router)#` **`distribute-list prefix FILTER out eigrp 10`**	Creates an outbound distribute list to filter routes being redistributed from EIGRP into OSPF that references the prefix list.
	NOTE: The implicit deny any statement at the end of the prefix list prevents routing updates about any other network from being advertised. As a result, networks 172.16.3.0/24 and 172.16.4.0/24 will not be redistributed into OSPF.

TIP: You can attach prefix lists to the redistribution process either via a distribute list or via a route map.

Verifying Prefix Lists

`show ip prefix-list` `[` **`detail`** `|` **`summary`**`]`	Displays information on all prefix lists. Specifying the **detail** keyword includes the description and the hit count (the number of times the entry matches a route) in the display.
`clear ip prefix-list` *`prefix-list-name`*`[network/length]`	Resets the hit count shown on prefix list entries.

Using Route Maps with Route Redistribution

`Router(config)#` **`route-map MY_MAP permit 10`**	Creates a route map called MY_MAP. This **route-map** statement will permit redistribution based on subsequent criteria. A sequence number of 10 is assigned.
`Router(config-route-map)#` **`match ip address 5`**	Specifies the match criteria (the conditions that should be tested); in this case, match addresses filtered using a standard access list number 5.
`Router(config-route-map)#` **`set metric 500`**	Specifies the set action (what action is to be performed if the match criteria is met); in this case, set the external metric to 500 (instead of the default value of 20).
`Router(config-route-map)#` **`set metric-type type-1`**	Specifies a second set action for the same match criteria. In this case, set the external OSPF network type to E1.

`Router(config-route-map)#route-map MY_MAP deny 20`	Adds a second statement to the MY_MAP route map that will deny redistribution based on subsequent criteria.
`Router(config-route-map)#match ip address prefix-list MY_PFL`	Specifies the match criteria (the conditions that should be tested); in this case, match addresses filtered using a prefix list named MY_PFL.
`Router(config-route-map)#route-map MY_MAP permit 30`	Adds a third statement to the MY_MAP route map that will permit redistribution based on subsequent criteria.
	NOTE: No "match" criteria are explicitly specified; all other routes will be redistributed with the following "set" criteria applied.
`Router(config-route-map)#set metric 5000`	Specifies the set action (what action is to be performed if the match criteria is met); in this case, set the external metric to 5000 (instead of the default value of 20)
`Router(config-route-map)#set metric-type type 2`	Specifies a second set action for the same match criteria; in this case, set the external OSPF network type to E2. This is optional since the default type for redistributed routes into OSPF is external type 2.
`Router(config-route-map)#router ospf 10`	Enters OSPF process ID 10 routing process.
`Router(config-router)#redistribute eigrp 1 route-map MY_MAP subnets`	Redistributes only EIGRP routes that are permitted by route map MY_MAP into OSPF.

NOTE: When used to filter redistribution, route map **permit** or **deny** statements determine whether the route will be redistributed. Routes without a match will not be redistributed. The route map stops processing at the first match (similar to an access list or prefix list). There is always an implicit deny statement at the end of a route map.

Configuration Example: Route Maps

Figure 4-8 shows the network topology for the configuration that follows, which demonstrates how to control redistribution with a route map using the commands covered in this chapter. Assume that all basic configurations and EIGRP and OSPF routing have been configured correctly.

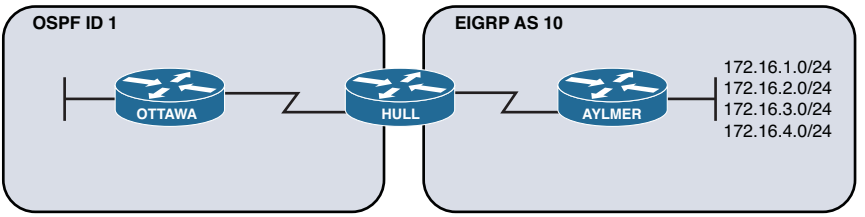


Figure 4-8 Network Topology for Route Map Configuration

The objective is to only redistribute networks 172.16.1.0/24 and 172.16.2.0/24 into OSPF and advertise them as external type 1 (E1) routes with an external metric of 50.

`HULL(config)#access-list 5 permit 172.16.1.0 0.0.0.255`	Creates a standard ACL number 5 and explicitly permits the 172.16.1.0/24 network.
`HULL (config)#access-list 5 permit 172.16.2.0 0.0.0.255`	Adds a second line to ACL 5 that explicitly permits the 172.16.2.0/24 network.
`HULL(config)#route-map FILTER permit 10`	Creates a route map called FILTER. This route map will permit traffic based on subsequent criteria. A sequence number of 10 is assigned.
`HULL(config-route-map)#match ip address 5`	Specifies the match criteria; match addresses filtered from ACL 5.
`HULL(config-route-map)#set metric 50` `HULL(config-route-map)#set metric-type type-1`	Specifies the set actions (what actions are to be performed if the match criterion is met); in this case, sets the external metric to 50 *and* sets the type to external type 1 (E1).
`HULL(config)#router ospf 1`	Enters OSPF process ID 1 routing process.
`HULL(config)#redistribute eigrp 10 subnets route-map FILTER`	Redistributes only those EIGRP networks into OSPF which match the route map.
	NOTE: Networks 172.16.2.0/24 and 172.16.3.0/24 will not be redistributed because of the implicit deny any at the end of the route map.

Manipulating Redistribution Using Route Tagging

Two-way multipoint redistribution can introduce routing loops in the network. One option to prevent redistribution of already redistributed routes is to use route tagging. In two-way multipoint redistribution scenarios, route tags must be applied and filtered in both direction and on both routers performing redistribution.

Figure 4-9 shows the network topology for the configuration that follows, which demonstrates how to control redistribution with route tags using the commands covered in this chapter. Assume that all basic configurations and EIGRP and OSPF routing have been configured correctly. A tag number of 11 is used to identify OSPF routes, and a tag of 22 is used to identify EIGRP routes.

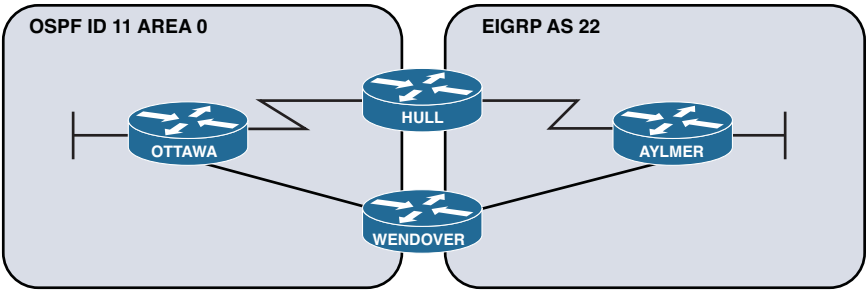


Figure 4-9 Network Topology for Redistribution Using Route Tagging

The following configuration needs to be entered on both the HULL and WENDOVER routers.

`HULL(config)#route-map EIGRPtoOSPF deny 10` `HULL(config-route-map)#match tag 11`	Creates a route map named EIGRPtoOSPF and denies redistribution for all routes tagged with the value 11.
`HULL(config-route-map)#route-map EIGRPtoOSPF permit 20` `HULL(config-route-map)#set tag 22`	Creates a second statement for route map EIGRPtoOSPF permitting all other routes to be redistributed with a tag of 22.
`HULL(config-route-map)#route-map OSPFtoEIGRP deny 10` `HULL(config-route-map)#match tag 22`	Creates a route map names OSPFtoEIGRP and denies redistribution for all routes tagged with the value 22.
`HULL(config-route-map)#route-map OSPFtoEIGRP permit 20` `HULL(config-route-map)#set tag 11`	Creates a second statement for route map OSPFtoEIGRP permitting all other routes to be redistributed with a tag of 11.
`HULL(config-route-map)#router ospf 11`	Enters OSPF configuration mode.
`HULL(config-router)#redistribute eigrp 22 subnets route-map EIGRPtoOSPF`	Redistributes all EIGRP routes with a tag of 22 into the OSPF domain.
`HULL(config-router)#router eigrp 22`	Enters EIGRP configuration mode.

`HULL(config-router)#redistribute ospf 11 metric 1500 1 255 1 1500 route-map OSPFtoEIGRP`	Redistributes all OSPF routes with a tag of 11 into the EIGRP domain.
	NOTE: The result here is to ensure only routes originating in the OSPF domain are redistributed into EIGRP, while only routes originating in the EIGRP domain are redistributed into the OSPF domain.

Changing Administrative Distance for Internal and External Routes

The commands to change the administrative distance (AD) for internal and external routes are as follows.

`Router(config)#router ospf 1`	Starts the OSPF routing process
`Router(config-router)#distance ospf intra-area 105 inter-area 105 external 125`	Changes the AD to 105 for intra-area and interarea routes, and changes the AD to 125 for external routes
`Router(config)#router eigrp 100`	Starts the EIGRP routing process
`Router(config-router)#distance eigrp 80 105`	Changes the AD to 80 for internal EIGRP routes and changes the AD to 105 for EIGRP external routes
`Router(config)#router bgp 65001`	Starts the BGP routing process
`Router(config-router)#distance bgp 30 200 220`	Changes the AD to 30 for external BGP routes, 200 for internal BGP routes and 220 for local BGP routes

Passive Interfaces

`Router(config)#router rip`	Starts the RIP routing process.
`Router(config-router)#passive-interface serial0/0/0`	Sets the interface as passive, meaning that routing updates will not be sent out this interface.
	NOTE: For RIP, the **passive-interface** command will prevent the interface from sending out routing updates but will allow the interface to receive updates.
`Router(config)#router rip`	Starts the RIP routing process.

`Router(config-router)#passive-interface default`	Sets all interfaces as passive.
	TIP: The **passive-interface default** command is useful for Internet service provider (ISP) and large enterprise networks, where a distribution router may have as many as 200 interfaces.
`Router(config-router)#no passive-interface fastethernet0/0`	Activates the FastEthernet0/0 interface to send and receive updates.

CAUTION: For OSPF, a passive interface does not send or process received Hellos. This prevents routers from becoming neighbors on that interface. A better way to control OSPF routing updates is to create a stub area, a totally stubby area, or a not-so-stubby area (NSSA).

CAUTION: When the **passive-interface** command is used with EIGRP, inbound and outbound hello packets are not sent. This prevents routers from becoming EIGRP neighbors. A passive interface cannot send EIGRP hellos, which prevents adjacency relationships with link partners. An administrator can create a "pseudo" passive EIGRP interface by using a route filter that suppresses all routes from the EIGRP routing update. An example of this is shown in Chapter 2, "EIGRP Implementation."

This page intentionally left blank

CHAPTER 5

Path Control Implementation

This chapter provides information about the following topics:

- Verifying Cisco Express Forwarding
- Configuring Cisco Express Forwarding
- Path control with policy-based routing
- Verifying policy-based routing
- Configuration example: PBR with route maps
- Cisco IOS IP service level agreements

Verifying Cisco Express Forwarding

`Router#show ip cef`	Displays a summary of the Cisco Express Forwarding Information Base (FIB) table. This information is derived from the routing table.
`Router#show adjacency`	Verifies that an adjacency exists for a connected device, that the adjacency is valid, and that the MAC header rewrite string is correct. This information is derived from the IP Address Resolution Protocol (ARP) table.
`Router#show ip route`	Displays the routing table.
`Router#show ip interface fastethernet0/0`	Verifies if CEF is enabled on the interface.

Configuring Cisco Express Forwarding

`Router(config)#no ip cef`	Disables CEF globally for IPv4. CEF is enabled by default.
`Router(config)#interface fastethernet0/0`	Enters interface FastEthernet0/0 configuration mode.
`Router(config-if)#no ip route-cache cef`	Disables CEF on the FastEthernet0/0 interface.

NOTE: CEF for IPv4 is enabled, by default, on all interfaces with the global-level **ip cef** command.

NOTE: CEF for IPv6, in contrast, is not enabled by default. However, it is enabled automatically when you enable IPv6 unicast routing. As a prerequisite, IPv4 CEF must be enabled in order to use IPv6 CEF. To disable IPv6 CEF, use the **no ipv6 cef** command.

Path Control with Policy-Based Routing

Path control is the mechanism that changes default packet forwarding across a network. It is not quality of service (QoS) or MPLS Traffic Engineering (MPLS-TE). Path control is a collection of tools or a set of commands that give you more control over routing by extending and complementing the existing mechanisms provided by routing protocols. Bypassing the default packet forwarding decision may be required to obtain better resiliency, performance, or availability in your network.

Configuring PBR is a two-step process. First, a route map is created which specifies the new forwarding decision to be implemented. Second, the route map is applied to an incoming interface.

`Router(config)#route-map ISP1 permit 10`	Creates a route map named ISP1. This route map will permit traffic based on subsequent criteria. A sequence number of 10 is assigned.
	NOTE: In route maps, the default action is to permit.
	NOTE: The *sequence-number* is used to indicate the position the route map statement is to have within the route map. A route map is comprised of route map statements with the same route map name.If no sequence number is given, the first statement in the route map is automatically numbered as 10.
`Router(config-route-map)#match ip address 1`	Specifies the match criteria (the conditions that should be tested); in this case, match addresses using ACL 1.
`Router(config-route-map)#set ip next hop 6.6.6.6`	Specifies the set actions (what action is to be performed if the match criteria are met); in this case, output packets to the router at IP address 6.6.6.6.
`Router(config-route-map)#set interface serial0/0/0`	Specifies the set actions (what action is to be performed if the match criteria are met); in this case, forward packets out interface Serial0/0/0.
	NOTE: If no explicit route exists in the routing table for the destination network address of the packet (that is, the packet is a broadcast packet or destined to an unknown address), the set interface command has no effect and is ignored.

	NOTE: A default route in the routing table will not be considered an explicit route for an unknown destination address.
`Router(config-route-map)#set ip default next hop 6.6.6.6`	Defines where to output packets that pass a match clause of a route map for policy routing and for which the router has no explicit route to the destination address.
`Router(config-route-map)#set default interface serial0/0/0`	Defines where to output packets that pass a match clause of a route map for policy routing and for which the router has no explicit route to the destination address.
	NOTE: This is recommended for point-to-point links only.
`Router(config-route-map)#exit`	Returns to global configuration mode.
`Router(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#ip policy route-map ISP1`	Specifies a route map to use for policy routing on an incoming interface that is receiving the packets that need to be policy routed.
`Router(config-if)#exit`	Returns to global configuration mode.
`Router(config)#ip local policy route-map ISP1`	Specifies a route map to use for policy routing on all packets originating on the router.

TIP: Packets that are generated by the router are not normally policy routed. Using **the ip local policy route-map** [*map-name*] command will make these packets adhere to a policy. For example, you may want packets originating at the router to take a route other than the best path according to the routing table.

Verifying Policy-Based Routing

`Router#show ip policy`	Displays route maps that are configured on the interfaces
`Router#show route-map [map-name]`	Displays route maps
`Router#debug ip policy`	Enables the display of IP policy routing events
`Router#traceroute`	Enables the extended **traceroute** command, which allows the specification of the source address
`Router#ping`	Enables the extended **ping** command, which allows for the specification of the source address

Configuration Example: PBR with Route Maps

Figure 5-1 shows the network topology for the configuration that follows, which demonstrates how to configure PBR with route maps using the commands covered in this chapter.

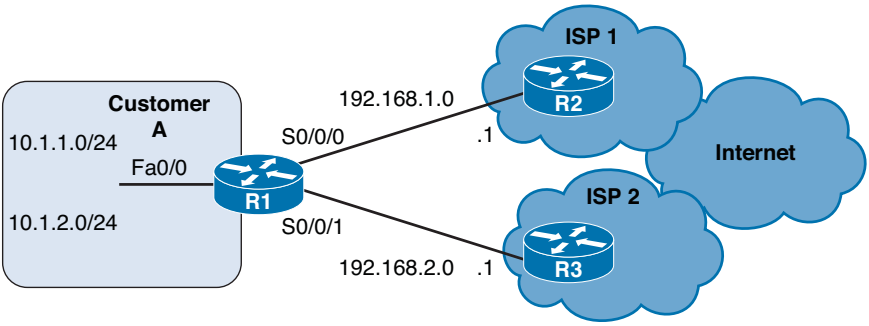


Figure 5-1 Network Topology for PBR with Route Maps

The objective is to forward Internet traffic sourced from the 10.1.1.0/24 network to ISP1 and traffic sourced from the 10.1.2.0/24 network to ISP2. Assume that all basic configurations and routing have been configured.

`R1(config)#access-list 11 permit 10.1.1.0 0.0.0.255`	Creates a standard access list that matches traffic originating from network 10.1.1.0/24. The number 11 is used for this ACL.
`R1(config)#access-list 12 permit 10.1.2.0 0.0.0.255`	Creates a standard access list that matches traffic originating from network 10.1.2.0/24. The number 12 is used for this ACL.
`R1(config)#route-map PBR permit 10`	Creates a route map named PBR. This route map will permit traffic based on subsequent criteria. A sequence number of 10 is assigned.
`R1(config-route-map) #match ip address 11`	Specifies the match criteria—match addresses permitted by ACL 11.
`R1(config-route-map)#set ip next-hop 192.168.1.1`	Specifies the set actions (what action is to be performed if the match criteria are met); in this case, forward packets to the router at 192.168.1.1 (ISP1).
`R1(config-route-map)#route-map PBR permit 20`	Adds a second statement to the PBR route map. A sequence number of 20 is assigned.
`R1(config-route-map)#match ip address 12`	Specifies the match criteria; match addresses permitted by ACL 12.

`R1(config-route-map)#set ip next-hop 192.168.2.1`	Specifies the set actions (what action is to be performed if the match criteria are met); in this case, forward packets to the router at 192.168.2.1 (ISP2).
`R1(config-route-map)#route-map PBR permit 30`	Adds a third statement to the PBR route map. A sequence number of 30 is assigned.
`R1(config-route-map)#set default interface null0`	Specifies that all other traffic not matching ACL 11 or ACL12 will be sent to the Null0 interface (traffic is dropped).
`R1(config-route-map)#exit`	Exits the route map configuration mode.
`R1(config)#interface fastethernet0/0`	Enters FastEthernet0/0 interface configuration mode.
`R1(config-if)#ip policy route-map PBR`	Applies the PBR route map to the interface. This is the incoming interface receiving the packets to be policy-routed.

Cisco IOS IP Service Level Agreements

NOTE: Cisco IOS IP service level agreements (SLAs) are used to perform network performance measurements within Cisco Systems devices using active traffic monitoring.

TIP: SLAs use time-stamp information to calculate performance metrics such as jitter, latency, network and server response times, packet loss, and mean opinion score.

Figure 5-2 shows the network topology for the configuration that follows, which shows the use of Cisco IOS IP SLA functionality for path control. Assume that all basic configurations have been configured.

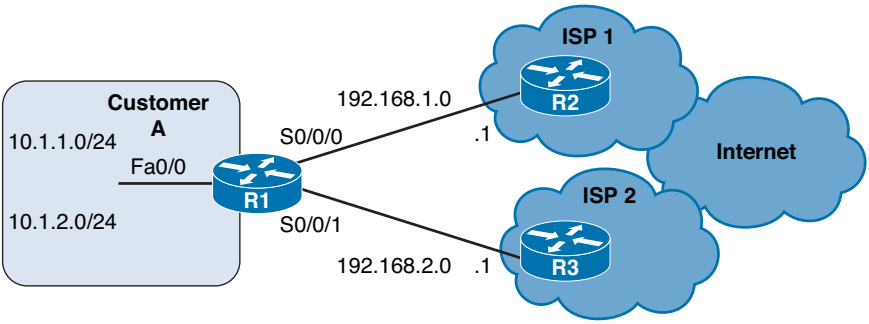


Figure 5-2 Network Topology for IOS IP SLA

Customer requirements:

Customer A is multihoming to ISP-1 and ISP-2.

The link to ISP-1 is the primary link for all traffic.

Customer A is using default routes to the Internet service providers (ISPs).

Customer A is using these default routes with different administrative distances to make ISP-1 the preferred route.

Potential problem: If ISP-1 is having uplink connectivity problems to the Internet, Customer A will still be sending all of its traffic to ISP-1, only to have that traffic lost.

Possible solutions: (1) IOS IP SLA will be used to announce conditionally the default route, *or* (2) the SLA will be used to verify availability for PBR.

Follow these steps to configure Cisco IOS IP SLA functionality:

1. Define one (or more) probe(s).
2. Define one (or more) tracking object(s).

3a. Define the action on the tracking object(s).

or

3b. Define policy routing using the tracking object(s).

4. Verify IP SLA operations.

NOTE: Only the configuration on R1 for neighbor ISP-1 is shown. Typically, in a multihoming scenario, R1 would be configured with two SLAs, two tracking objects, and two default routes.

Step 1: Define One (or More) Probe(s)

`R1(config)#ip sla 1`	Begins configuration for an IP SLA operation and enters SLA configuration mode. 1 is the operation number and can be a number between 1 and 2,147,483,647.
`R1(config-ip-sla)#icmp-echo 192.168.1.1 source-interface fastethernet0/0`	Defines an ICMP echo operation to destination address 192.168.1.1using a source interface of FastEthernet0/0 and enters ICMP echo configuration mode.
	TIP: Typically, the address tested is within the ISP network instead of the next hop.
`R1(config-ip-sla-echo)#frequency 10`	Sets the rate at which the operation repeats. Measured in seconds from 1 to 604,800 (7 days).

`R1(config-ip-sla-echo)#timeout 5000`	Length of time the operation waits to receive a response from its request packet, in milliseconds. Range is 0 to 604,800,000.
	TIP: It is recommended that the timeout value be based on the sum of both the maximum round-trip time (RTT) value for the packets and the processing time of the IP SLAs operation.
`R1(config-ip-sla-echo)#exit`	Exits IP SLA ICMP echo configuration mode and returns to global configuration mode.
`R1(config)#ip sla schedule 1 start-time now life forever`	Sets a schedule for IP SLA monitor 1. Packets will be sent out immediately and will continue forever.

Step 2: Define One (or More) Tracking Object(s)

`R1(config)#track 11 ip sla 1 reachability`	Configures the tracking process to track the reachability of IP SLAs operation 11. The number 1 refers to the SLA defined in Step 1.

Step 3a: Define the Action on the Tracking Object(s)

`R1(config)#ip route 0.0.0.0 0.0.0.0 192.168.1.1 2 track 11`	Announces a default route to 192.168.1.1 with an administrative distance of 2 if tracking object 11 is true.

Or

Step 3b: Define Policy Routing Using the Tracking Object(s)

`R1(config)#route-map IPSLA permit 10`	Creates a route map which will use the tracking object.
`R1(config-route-map)#set ip next-hop verify-availability 192.168.1.1 10 track 11`	Configures policy routing to verify the reachability of the next hop 192.168.1.1 before the router performs policy routing to that next hop. A sequence number of 10 is used and tracking object 11 is referenced.
	NOTE: The sequence number is used when tracking the availability of multiple addresses. Each address tracked would get its own sequence number (for example, 10, 20, 30). If the first tracking objects fails, the next one in the sequence is used. If all tracking objects fail, the policy routing fails, and the packets are routed according to the routing table.
	TIP: Typically, the address tested is within the ISP network instead of the next hop.

`R1(config-route-map)#interface fastethernet0/0`	Enters interface configuration mode.
`R1(config-if)#ip policy route-map IPSLA`	Applies the IPSLA route map to the interface. This is the incoming interface receiving the packets to policy routed.

Step 4: Verify IP SLA Operations

`R1#show ip sla configuration`	Displays configuration values including all defaults for all SLAs
`R1#show ip sla statistics`	Displays the current operational status and statistics of all SLAs
`R1#show track`	Displays information about objects that are tracked by the tracking process

NOTE: Effective with Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the **ip sla monitor** command is replaced by the **ip sla** command.

NOTE: Effective with Cisco IOS Release 12.4(4)T, 12.2(33)SB, and 12.2(33)SXI, the **type echo protocol ipIcmpEcho** command is replaced by the **icmp-echo** command.

NOTE: Effective with Cisco IOS Release 12.4(20)T, 12.2(33)SXI1, 12.2(33)SRE and Cisco IOS XE Release 2.4, the **track rtr** command is replaced by the **track ip sla** command.

NOTE: Effective with Cisco IOS Release 12.4(20)T, 12.2(33)SXI1, 12.2(33)SRE, and Cisco IOS XE Release 2.4, the **show ip sla monitor configuration** command is replaced by the **show ip sla configuration** command.

NOTE: Effective with Cisco IOS Release 12.4(20)T, 12.2(33)SXI1, 12.2(33)SRE, and Cisco IOS XE Release 2.4, the **show ip sla monitor statistics** command is replaced by the **show ip sla statistics** command.

CHAPTER 6

Enterprise Internet Connectivity

This chapter provides information about the following topics:

- Configuring a provider-assigned static or DHCP IPv4 address
- Configuring static NAT
- Configuring dynamic NAT
- Configuring NAT overload (PAT)
- Verifying NAT
- NAT virtual interface
- Configuration example: NAT virtual interfaces and static NAT
- Configuring basic IPv6 Internet connectivity
- Configuring IPv6 ACLs
- Verifying IPv6 ACLs
- Configuring conditional redistribution of a default route in a dual-homed Internet connectivity scenario
- Configuring BGP
- BGP and loopback addresses
- iBGP next-hop behavior
- eBGP multihop
- Verifying BGP connections
- Troubleshooting BGP connections
- Default routes
- Attributes
    - Route selection decision process
    - Weight attribute
    - Using AS_PATH access lists to manipulate the weight attribute
    - Using prefix lists and route maps to manipulate the weight attribute
    - Local preference attribute
    - Using AS_PATH access lists and route maps to manipulate the local preference attribute
    - AS_PATH attribute prepending
    - AS_PATH: removing private autonomous systems
    - MED attribute

- Route aggregation
- Route reflectors
- Regular expressions
- Regular expressions: examples
- Configuration example: using prefix lists and AS_PATH access lists
- BGP peer groups
- MP-BGP
    - Configuring MP-BGP using address families to exchange IPv4 and IPv6 routes
    - Verifying MP-BGP

Configuring a Provider Assigned Static or DHCP IPv4 Address

Figure 6-1 shows the network topology for the configuration that follows, which demonstrates how to configure a provider assigned static IPv4 address or a provider assigned IPv4 DHCP address.

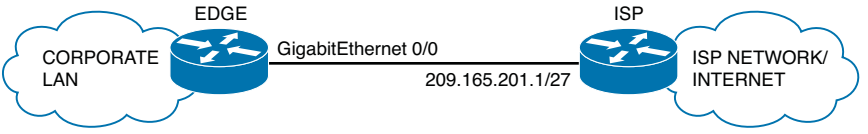


Figure 6-1 Configure a Provider Assigned Static or DHCP IPv4 Address

`EDGE(config)#interface gigabitethernet0/0`	Enters GigabitEthernet0/0 interface configuration mode
`EDGE(config-if)#ip address 209.165.201.2 255.255.255.224`	Assigns a static IPv4 address
`EDGE(config-if)#no shutdown`	Enables the interface
`EDGE(config-if)#ip route 0.0.0.0 0.0.0.0 209.165.201.1`	Defines a default route to the Internet service provider (ISP) next-hop IP address of 209.165.201.1

Or

`EDGE(config)#interface gigabitethernet0/0`	Enters GigabitEthernet0/0 interface configuration mode
`EDGE(config-if)#ip address dhcp`	Allows the interface to obtain an address dynamically from the ISP
`EDGE(config-if)#no shutdown`	Enables the interface

NOTE: If the default gateway optional parameter is contained within the Dynamic Host Configuration Protocol (DHCP) reply packet, the router will install a static default route in its routing table, with the default gateway's IP address as the next hop. The default route is installed with the administrative distance of 254, which makes it a floating static route. To disable this feature, use the interface-level command **no ip dhcp client request router**.

Configuring Static NAT

Figure 6-2 shows the network topology for the configuration that follows, which demonstrates how to configure static Network Address Translation (NAT). The objective here is to statically translate the address of the server to a public IP address.

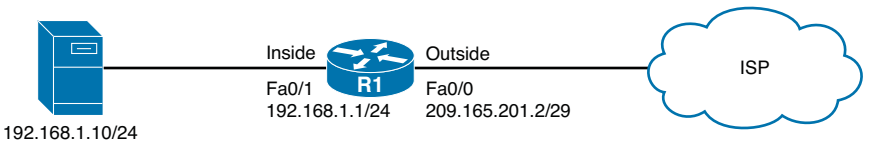


Figure 6-2 Configuring Static NAT

`R1(config)#interface fastethernet0/0`	Enters FastEthernet0/0 interface configuration mode.
`R1(config-if)#ip address 209.165.201.2 255.255.255.248`	Assigns a public IP address to the outside interface.
`R1(config-if)#ip nat outside`	Defines which interface is the outside interface for NAT.
`R1(config-if)#interface fastethernet0/1`	Enters FastEthernet0/1 interface configuration mode.
`R1(config-if)#ip address 192.168.1.1 255.255.255.0`	Assigns a private IP address to the inside interface.
`R1(config-if)#ip nat inside`	You can have more than one NAT inside interface on a router.
`R1(config-if)#exit`	Returns to global configuration mode.
`R1(config)#ip nat inside source static 192.168.1.10 209.165.201.5`	Permanently translates the inside address of 192.168.1.10 to a public address of 209.165.201.5. Use the command for each of the private IP addresses you want to statically map to a public address.

Configuring Dynamic NAT

Figure 6-3 shows the network topology for the configuration that follows, which demonstrates how to configure dynamic NAT. The objective here is to dynamically translate the addresses of the PCs to a range of public IP addresses.

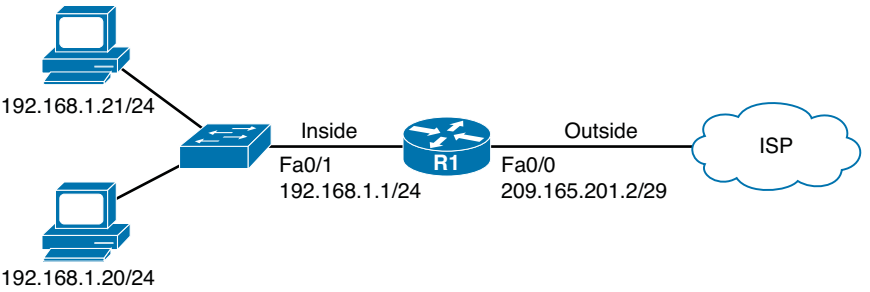


Figure 6-3 Configuring Dynamic NAT

`R1(config)#access-list 1 permit 192.168.1.0 0.0.0.255`	Defines an access list that identifies the private network that will be translated.
`R1(config)#ip nat pool R1_POOL 209.165.201.8 209.165.201.15 netmask 255.255.255.248`	Creates a pool of eight public addresses named R1_POOL that will be used for translation.
`R1(config)#interface fastethernet0/0`	Enters FastEthernet0/0 interface configuration mode
`R1(config-if)#ip address 209.165.201.2 255.255.255.248`	Assigns a public IP address to the outside interface.
`R1(config-if)#ip nat outside`	Defines which interface is the outside interface for NAT.
`R1(config-if)#interface fastethernet0/1`	Enters FastEthernet0/1 interface configuration mode.
`R1(config-if)#ip address 192.168.1.1 255.255.255.0`	Assigns a private IP address to the inside interface.
`R1(config-if)#ip nat inside`	You can have more than one NAT inside interface on a router.
`R1(config-if)#exit`	Returns to global configuration mode.
`R1(config)#ip nat inside source list 1 pool R1_POOL`	Enables translation of addresses permitted by ACL number 1 to the addresses in pool R1_POOL.

Configuring NAT Overload (PAT)

Figure 6-4 shows the network topology for the configuration that follows, which demonstrates how to configure NAT overload or Port Address Translation (PAT). The objective here is to translate the PC's addresses to the address of the router's public interface.

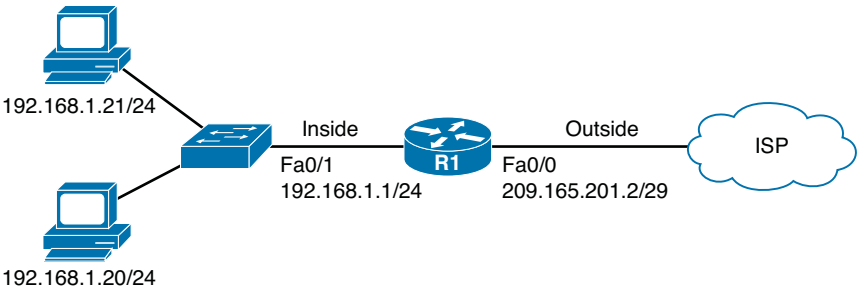


Figure 6-4 Configuring NAT Overload (PAT)

`R1(config)#access-list 1 permit 192.168.1.0 0.0.0.255`	Defines an access list that identifies the private network which will be translated.
`R1(config)#interface fastethernet0/0`	Enters FastEthernet0/0 interface configuration mode.
`R1(config-if)#ip address 209.165.201.2 255.255.255.248`	Assigns a public IP address to the outside interface.
`R1(config-if)#ip nat outside`	Defines which interface is the outside interface for NAT.
`R1(config-if)#interface fastethernet0/1`	Enters FastEthernet0/1 interface configuration mode.
`R1(config-if)#ip address 192.168.1.1 255.255.255.0`	Assigns a private IP address to the inside interface.
`R1(config-if)#ip nat inside`	You can have more than one NAT inside interface on a router.
`R1(config-if)#exit`	Returns to global configuration mode.
`R1(config)#ip nat inside source list 1 interface fasthethernet0/0 overload`	Enables translation of addresses permitted by ACL number 1 and uses the interface FastEthernet0/0 IP address for the NAT process. The keyword **overload** allows multiple inside devices to share a single public IP address while keeping track of port numbers to ensure sessions remain unique.

NOTE: It is possible to overload a dynamic pool instead of an interface. This allows the inside private devices to share multiple public IP address instead of only one. Use the command **ip nat inside source list** *acl* **pool** *pool* **overload** to achieve this.

Verifying NAT

`R1#show ip nat translation`	Displays the protocol, the inside global, inside local, outside local, and outside global addresses used in translation
`R1#show ip nat statistics`	Displays NAT statistics

NAT Virtual Interface

NAT virtual interface, or NVI, removes the requirements to configure an interface as either inside or outside. Also, because NVI performs routing, translation, and routing again, it is possible to route packets from inside to inside interfaces successfully.

`R1(config-if)#ip nat enable`	Allows the interface to participate in NVI translation processing.
`R1#show ip nat nvi translations`	Displays the list of active NVI translations.
	NOTE: Legacy NAT terminology does not apply because there are no "inside" or "outside" interfaces. Instead, NVI uses the source global, source local, destination global, and destination local terminology.
`R1#show ip nat nvi statistics`	Displays the interfaces participating in NVI translation processing, as well as Hit and Miss counters.

Configuration Example: NAT Virtual Interfaces and Static NAT

Figure 6-5 shows the network topology for the configuration that follows, which demonstrates how to configure NAT virtual interfaces with dynamic NAT and static NAT, using the commands covered in this chapter. Assume that all basic configurations are accurate.

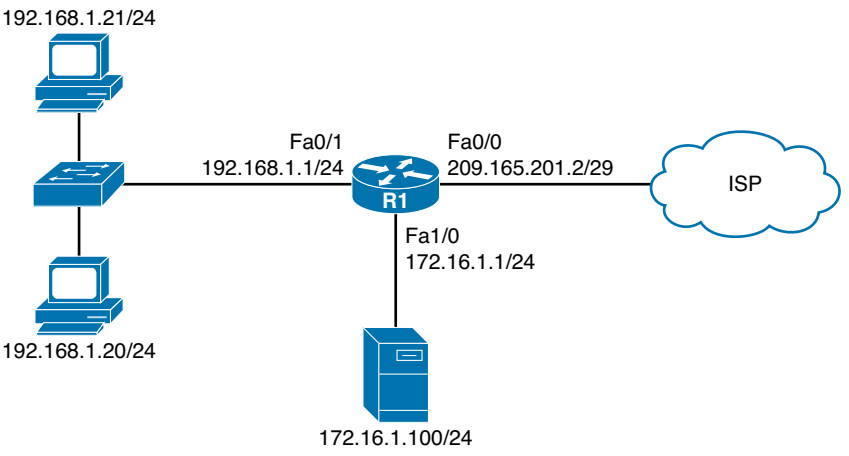


Figure 6-5 Configuration Example: NAT Virtual Interfaces and Static NAT

`R1(config)#access-list 1 permit 192.168.1.0 0.0.0.255`	Defines an access list that identifies the private network that will be translated
`R1(config)#ip nat pool R1_POOL 209.165.201.8 209.165.201.15 netmask 255.255.255.248`	Creates a pool of eight public addresses named R1_POOL that will be used for translation
`R1(config)#ip nat source list 1 pool R1_POOL`	Enables translation of address permitted by ACL number 1 to the addresses in pool R1_POOL
`R1(config)#ip nat source static 172.16.1.100 209.165.201.5`	Permanently translates the inside address of 172.16.1.100 to a public address of 209.165.201.5
`R1(config)#interface fastethernet0/0`	Enters FastEthernet0/0 interface configuration mode
`R1(config-if)#ip nat enable`	Enables NVI processing on the interface
`R1(config-if)#interface fastethernet0/1`	Enters FastEthernet0/1 interface configuration mode
`R1(config-if)#ip nat enable`	Enables NVI processing on the interface
`R1(config-if)#interface fastethernet1/0`	Enters FastEthernet1/0 interface configuration mode
`R1(config-if)#ip nat enable`	Enables NVI processing on the interface

Configure Basic IPv6 Internet Connectivity

Figure 6-6 shows the network topology for the configuration that follows, which demonstrates how to configure basic IPv6 Internet connectivity. Assume that all basic configurations are accurate.

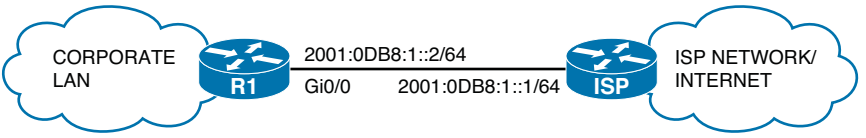


Figure 6-6 Configure Basic IPv6 Connectivity

`R1(config)#interface gigabitethernet0/0`	Enters GigabitEthernet0/0 interface configuration mode.
`R1(config-if)#ipv6 address 2001:0DB8:1::2/64`	Assigns a provider assigned IPv6 address.
`R1(config-if)#no shutdown`	Enables the Gigabit Ethernet 0/0 interface.
`R1(config-if)#exit`	Returns to global configuration mode.
`R1(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router.
`R1(config)#ipv6 route ::/0 2001:0DB8:1::1`	Creates a default static route pointing to the ISP's next-hop IPv6 address. All nonlocal traffic will be forwarded to the ISP.

Configuring IPv6 ACLs

Figure 6-7 shows the network topology for the configuration that follows, which demonstrates how to configure IPv6 ACLs. Assume that all basic configurations are accurate. The objective here is to create an ACL that will act as a firewall allowing HTTP, HTTPS, DNS, and ICMP traffic to return from the Internet.

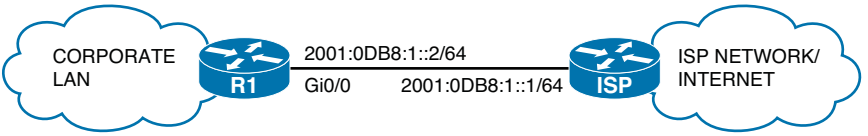


Figure 6-7 Configure IPv6 ACLs

`R1(config)#ipv6 access-list FIREWALL`	Creates a named extended IPv6 access list called FIREWALL and moves to IPv6 access list configuration mode.
`R1(config-ipv6-acl)#permit tcp any eq www any established`	Permits HTTP traffic to return to the corporate LAN from the Internet if that traffic was originally sourced from the corporate LAN.
`R1(config-ipv6-acl)#permit tcp any eq 443 any established`	Permits HTTPS traffic to return to the corporate LAN from the Internet if that traffic was originally sourced from the corporate LAN.
`R1(config-ipv6-acl)#permit udp any eq domain any`	Permits DNS responses to return to the corporate LAN from the Internet.
`R1(config-ipv6-acl)#permit icmp any any echo-reply`	Permits ICMP ping responses to return to the corporate LAN from the Internet.
`R1(config-ipv6-acl)#permit icmp any any packet-too-big`	Permits ICMP Packet Too Big messages to return to the corporate LAN from the Internet.
	NOTE: In IPv6, maximum transmission unit (MTU) discovery has moved from the router to the hosts. It is important to allow Packet Too Big messages to flow through the router to allow hosts to detect whether fragmentation is required.
`R1(config-ipv6-acl)#exit`	Returns to global configuration mode.
`R1(config)#interface gigabitethernet0/0`	Enters GigabitEthernet0/0 interface configuration mode.
`R1(config-if)#ipv6 traffic-filter FIREWALL in`	Applies the IPv6 access list names FIREWALL to the interface in the inbound direction.

NOTE: The "implicit deny" rule has changed for IPv6 access lists to take into account the importance of the Neighbor Discovery Protocol (NDP). NDP is to IPv6 what Address Resolution Protocol (ARP) is to IPv4, so naturally the protocol should not be disrupted. That is the reason two additional implicit statements have been added before the "implicit deny" statement at the end of each IPv6 ACL.

These implicit rules are as follows:

```
permit icmp any any nd-na
permit icmp any any nd-ns
```

deny ipv6 any anyIt is important to understand that any explicit **deny ipv6 any any** statement overrides all three implicit statements, which can lead to problems because NDP traffic is blocked.

Verifying IPv6 ACLs

`R1#show ipv6 access-list`	Displays the configured statements, their matches, and sequence number of all access lists

Configuring Redistribution of Default Routes with Different Metrics in a Dual-Homed Internet Connectivity Scenario

Figure 6-8 shows the network topology for the configuration that follows, which demonstrates how to configure redistribution of default routes with difference metrics. Assume that all basic configurations are accurate. The objective here is to redistribute two default routes, one used for the primary link to the ISP and one used for the backup link to the same ISP. The metric values are manipulated to make the primary link the preferred route.

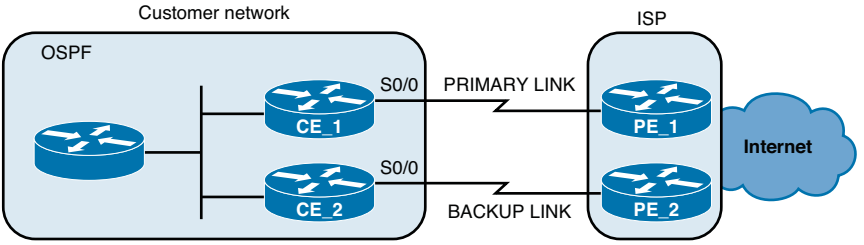


Figure 6-8 Configure Redistribution of Default Routes with Different Metrics in a Dual-Homed Internet Connectivity Scenario

`CE_1(config)#ip route 0.0.0.0 0.0.0.0 serial0/0`	Creates a default static route to the ISP's PE_1 router
`CE_1(config)#router ospf 1`	Enters the OSPF routing process

`CE_1(config-router)#redistribute static metric-type 1 metric 100`	Redistributes the default route into OSPF as an external type 1 (E1) route with an initial seed metric of 100
`CE_2(config)#ip route 0.0.0.0 0.0.0.0 serial0/0`	Creates a default static route to the ISP's PE_2 router
`CE_2(config)#router ospf 1`	Enters the OSPF routing process
`CE_2(config-router)#redistribute static metric-type 1 metric 200`	Redistributes the default route into OSPF as an external type 1 (E1) route with an initial seed metric of 200

Configuring BGP

`Router(config)#router bgp 100`	Starts BGP routing process 100.
	NOTE: Cisco IOS software permits only one Border Gateway Protocol (BGP) process to run at a time; therefore, a router cannot belong to more than one autonomous system.
`Router(config-router)# neighbor 192.31.7.1 remote-as 200`	Identifies a peer router with which this router will establish a BGP session. The autonomous system number will determine whether the neighbor router is an external BGP (eBGP) or internal BGP (iBGP) neighbor.
	TIP: If the autonomous system number configured in the **router bgp** command is identical to the autonomous system number configured in the **neighbor** statement, BGP initiates an internal session (iBGP). If the field values differ, BGP builds an external session (eBGP).
	TIP: **neighbor** statements must be symmetrical for a neighbor relationship to be established.
`Router(config-router)# network 192.135.250.0`	Tells the BGP process what locally learned networks to advertise.
	NOTE: The networks can be connected routes, static routes, or routes learned via a dynamic routing protocol, such as Open Shortest Path First (OSPF) Protocol.
	NOTE: Configuring just a **network** statement will not establish a BGP neighbor relationship.
	NOTE: The networks must also exist in the local router's routing table; otherwise, they will not be sent out in updates.
`Router(config-router)# network 128.107.0.0 mask 255.255.255.0`	Used to specify an individual subnet which must be present in the routing table or it will not be advertised by BGP.

TIP: Routes learned by the BGP process are propagated by default but are often filtered by a routing policy.

CAUTION: If you misconfigure a **network** command, such as the example **network 192.168.1.1 mask 255.255.255.0**, BGP will look for exactly 192.168.1.1/24 in the routing table. It may find 192.168.1.0/24 or 192.168.1.1/32; however, it may never find 192.168.1.1/24. Because there is no match for the network, BGP does not announce it to any neighbors.

TIP: If you issue the command **network 192.168.0.0 mask 255.255.0.0** to advertise a CIDR block, BGP will look for 192.168.0.0/16 in the routing table. It may find 192.168.1.0/24 or 192.168.1.1/32; however, it may never find 192.168.0.0/16. Because there is no match to the network, BGP does not announce this network to any neighbors. In this case, you can configure a static route towards a null interface so BGP can find an exact match in the routing table:

ip route 192.168.0.0 255.255.0.0 null0

After finding this exact match in the routing table, BGP will announce the 192.168.0.0/16 network to any neighbors.

BGP and Loopback Addresses

`Router(config)#router bgp 100`	Starts the BGP routing process.
`Router(config-router)#` **`neighbor 172.16.1.2 update-source loopback0`**	Informs the router to use any operational interface as the source IP address for TCP connections (in this case, Loopback0). Because a loopback interface never goes down, this adds more stability to your configuration as compared to using a physical interface.
	TIP: Without the **neighbor update-source** command, BGP will use the closest IP interface to the peer. This command provides BGP with a more robust configuration, because BGP will still operate in the event the link to the closest interface fails.
	NOTE: You can use the **neighbor update-source** command with either eBGP or iBGP sessions. In the case of a point-to-point eBGP session, this command is not needed because there is only one path for BGP to use.

iBGP Next-Hop Behavior

The eBGP next-hop attribute is the IP address that is used to reach the advertising router. For eBGP peers, the next-hop address is, in most cases, the IP address of the connection between the peers. For iBGP, the eBGP next-hop address is carried into the local autonomous system.

Figure 6-9 shows the network topology for the configuration that follows, which demonstrates how to configure the next-hop attribute. The objective here is to allow R3 to learn the correct next-hop address when trying to reach networks outside its autonomous system. Assume that all basic and OSPF configurations are accurate.

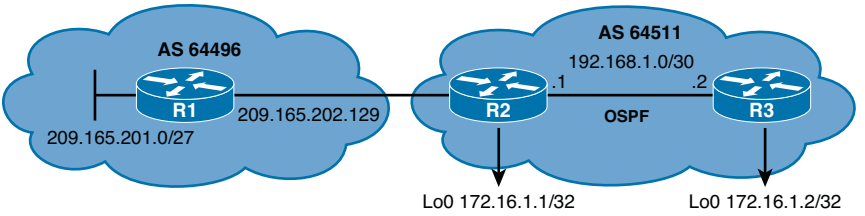


Figure 6-9 iBGP Next-Hop Behavior

`R2(config)#router bgp 64511`	Starts the BGP routing process.
`R2(config-router)#neighbor 209.165.202.129 remote-as 64496`	Identifies R1 as an eBGP neighbor.
`R2(config-router)#neighbor 172.16.1.2 remote-as 64511`	Identifies R3 as an iBGP neighbor.
`R2(config-router)#neighbor 172.16.1.2 update-source loopback0`	Informs R2 to use Loopback0 IP address (172.16.1.1) as the source IP address for all BGP TCP packets sent to R3.
`R2(config-router)#neighbor 172.16.1.2 next-hop-self`	Allows R2 to advertise itself as the next hop to its iBGP neighbor for networks learned from autonomous system 64496. R3 will then use 172.16.1.1 as the next hop to reach network 209.165.201.0/27 instead of using the eBGP next-hop of 209.165.202.129.

eBGP Multihop

By default, eBGP neighbors exchange packets with a TTL (Time To Live) set to 1. If you attempt to establish eBGP session between loopbacks, BGP packets will be dropped due to an expired TTL.

Figure 6-10 shows the network topology for the configuration that follows, which demonstrates how to configure eBGP multihop. Assume that all basic configurations are accurate.

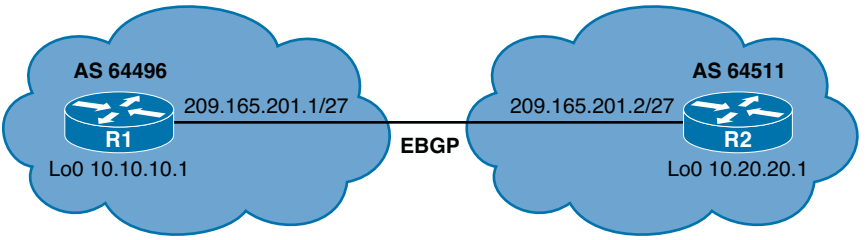


Figure 6-10 eBGP Multihop

`R1(config)#ip route 10.20.20.1 255.255.255.255 209.165.201.2`	Defines a static route to the Loopback 0 address on R2.
`R1(config)#router bgp 64496`	Starts the BGP routing process.
`R1(config-router)#neighbor 10.20.20.1 remote-as 64511`	Identifies a peer router at 10.20.20.1
`R1(config-router)#neighbor 10.20.20.1 update-source loopback0`	Informs R1 to use Loopback0 IP address as the source IP address for all BGP TCP packets sent to R2.
`R1(config-router)#neighbor 10.20.20.1 ebgp-multihop 2`	Allows for two routers that are not directly connected to establish an eBGP session. A TTL value of 2 is defined.
`R2(config)#ip route 10.10.10.1 255.255.255.255 209.165.201.1`	Defines a static route to the Loopback 0 address on R1.
`R2(config)#router bgp 64511`	Starts the BGP routing process.
`R2(config-router)#neighbor 10.10.10.1 remote-as 64496`	Identifies a peer router at 10.10.10.1
`R2(config-router)#neighbor 10.10.10.1 update-souce loopback0`	Informs R2 to use Loopback0 IP address as the source IP address for all BGP TCP packets sent to R1.
`R2(config-router)#neighbor 10.10.10.1 ebgp-multihop 2`	Allows for two routers that are not directly connected to establish an eBGP session. A TTL value of 2 is defined.

NOTE: The **ebgp-multihop** keyword is a Cisco IOS option. It must be configured on each peer. The **ebgp-multihop** keyword is only used for eBGP sessions, not for iBGP.eBGP neighbors are usually directly connected (over a WAN connection, for example) to establish an eBGP session. However, sometimes one of the directly connected routers is unable to run BGP. The **ebgp-multihop** keyword allows for a logical connection to be made between peer routers, even if they are not directly connected. The **ebgp-multihop** keyword allows for an eBGP peer to be up to 255 hops away and still create an eBGP session.

NOTE: If redundant links exist between two eBGP neighbors and loopback addresses are used, you must configure **ebgp-multihop** because of the default TTL of 1. Otherwise, the router decrements the TTL before giving the packet to the loopback interface, meaning that the normal IP forwarding logic discards the packet.

Verifying BGP Connections

`Router#show ip bgp`	Displays entries in the BGP table
`Router#show ip bgp neighbors`	Displays information about the BGP and TCP connections to neighbors
`Router#show ip bgp rib-failure`	Displays networks that are not installed in the Routing Information Base (RIB) and the reason that they were not installed
`Router#show ip bgp summary`	Displays the status of all BGP connections
`Router#show ip route bgp`	Displays the BGP entries from the routing table

Troubleshooting BGP Connections

`Router#clear ip bgp *`	Forces BGP to clear its table and resets all BGP sessions.
`Router#clear ip bgp 10.1.1.1`	Resets the specific BGP session with the neighbor at 10.1.1.1.
`Router#clear ip bgp 10.1.1.2 soft out`	Forces the remote router to resend all BGP information to the neighbor without resetting the connection. Routes from this neighbor are not lost.
	TIP: The **clear ip bgp w.x.y.z soft out** command is highly recommended when you are changing an outbound policy on the router. The **soft out** option does not help if you are changing an inbound policy.
	TIP: The **soft** keyword of this command is optional; **clear ip bgp out** will do a soft reset for all outbound updates.
`Router(config-router)# neighbor 10.1.1.2 soft-reconfiguration inbound`	Causes the router to store all updates from this neighbor in case the inbound policy is changed.
	CAUTION: The **soft-reconfiguration inbound** command is memory intensive.
`Router#clear ip bgp 10.1.1.2 soft in`	Uses the stored information to generate new inbound updates.
`Router#clear ip bgp {*|10.1.1.2} [soft in | in]`	Creates a dynamic soft reset of inbound BGP routing table updates. Routes are not withdrawn. Updates are not stored locally. The connection remains established. See the note that follows for more information on when this command can be used.

NOTE: Beginning with Cisco IOS Releases 12.0(2)S and 12.0(6)T, Cisco introduced a BGP soft reset enhancement feature known as *route refresh*. Route refresh is not dependent on stored routing table update information. This method requires no pre-configuration and requires less memory than previous soft methods for inbound routing table updates.

NOTE: To determine whether a BGP router supports route refresh capability, use the **show ip bgp neighbors** command. The following message is displayed in the output when route refresh is supported:

```
Received route refresh capability from peer
```

NOTE: When a BGP session is reset and soft reconfiguration is used, several commands enable you to monitor BGP routes that are received, sent, or filtered:

```
Router#show ip bgp
Router#show ip bgp neighbor address advertised
Router#show ip bgp neighbor address received
Router#show ip bgp neighbor address routes
```

`Router#debug ip bgp`	Displays information related to processing BGP
`Router#debug ip bgp updates`	Displays information about the processing of BGP update

CAUTION: The **clear ip bgp *** command is both processor and memory intensive and should be used only in smaller environments. A more reasonable approach is to clear only a specific network or a specific session with a neighbor with the **clear ip bgp** *specific-network* command. However, you can use this command whenever the following changes occur:

- Additions or changes to the BGP-related access lists
- Changes to BGP-related weights
- Changes to BGP-related distribution lists
- Changes in the BGP timer's specifications
- Changes to the BGP administrative distance
- Changes to BGP-related route maps

Default Routes

`Router(config)#router bgp 100`	Starts the BGP routing process
`Router(config-router)#neighbor 192.168.100.1 remote-as 200`	Identifies a peer router at 192.168.100.1
`Router(config-router)#neighbor 192.168.100.1 default-originate`	States that the default route of 0.0.0.0 will only be sent to 192.168.100.1

NOTE: If you want your BGP router to advertise a default to all peers and the 0.0.0.0 route exists in the routing table, use the network command with an address of 0.0.0.0:

```
R1(config)#router bgp 100
R1(config-router)#neighbor 172.16.20.1 remote-as 150
R1(config-router)#neighbor 172.17.1.1 remote-as 200
R1(config-router)#network 0.0.0.0
```

Attributes

Routes learned via BGP have associated properties that are used to determine the best route to a destination when multiple paths exist to a particular destination. These properties are referred to as *BGP attributes*, and an understanding of how BGP attributes influence route selection is required for the design of robust networks. After describing the route selection process, this section describes the attributes that BGP uses in the route selection process.

Route Selection Decision Process

Initially, a path is not considered if its next hop cannot be reached. Afterward, the decision process for determining the best path to reach a destination is based on the following:

1. Prefer the path with the *highest weight* (local to the router).
2. If the weights are the same, prefer the path with the *highest local preference* (global within the autonomous system).
3. If the local preferences are the same, prefer the path that was *originated by the local router* (next-hop = 0.0.0.0).
4. If no route was originated, prefer the route that has the *shortest autonomous system path*.
5. If all paths have the same autonomous system path length, prefer the path with the *lowest origin* code (where IGP is lower than EGP, and EGP is lower than Incomplete).
6. If the origin codes are the same, prefer the path with the *lowest MED attribute*.
7. If the paths have the same MED, prefer the *external path* over the internal path.
8. If the paths are still the same, prefer the path through the *closest IGP neighbor*.
9. For eBGP paths, select the *oldest route* to minimize the effects of route flapping.
10. Prefer the route with the *lowest neighbor BGP router ID value*.
11. If the BGP router IDs are the same, prefer the router with the *lowest neighbor IP address*.

Weight Attribute

The weight is configured locally on a router and is not propagated to any other routers. This attribute applies when one router is used with multiple exit points out of an

autonomous system, as opposed to the local preference attribute, which is used when two or more routers provide multiple exit points.

Figure 6-11 shows the network topology for the configuration that follows, which demonstrates how to configure the weight attribute. Assume that all basic configurations are accurate.

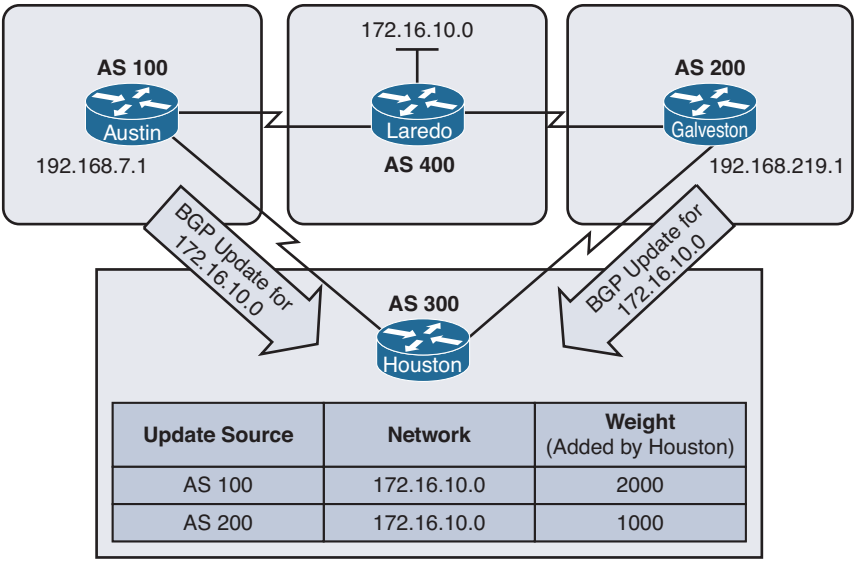


Figure 6-11 Weight Attribute

`Houston(config)#router bgp 300`	Starts the BGP routing process
`Houston(config-router)#neighbor 192.168.7.1 remote-as 100`	Identifies a peer router at 192.168.7.1
`Houston(config-router)#neighbor 192.168.7.1 weight 2000`	Sets the weight of all route updates from neighbor 192.168.7.1 to 2000
`Houston(config-router)#neighbor 192.168.219.1 remote-as 200`	Identifies a peer router at 192.168.219.1
`Houston(config-router)#neighbor 192.168.219.1 weight 1000`	Sets the weight of all route updates from neighbor 192.168.219.1 to 1000

The result of this configuration will have Houston forward traffic to the 172.16.10.0 network through autonomous system 100, because the route entering autonomous system 300 from autonomous system 100 had a higher **weight** attribute set compared to that same route advertised from autonomous system 200.

NOTE: The **weight** attribute is local to the router and not propagated to other routers. By default, the weight attribute is 32,768 for paths that the router originates, and 0 for other paths. Routes with a *higher weight are preferred* when there are multiple routes to the same destination.

Using AS_PATH Access Lists to Manipulate the Weight Attribute

Refer to Figure 6-11 for the configuration that follows, which demonstrates how to configure the weight attribute using AS_PATH access lists.

`Houston(config)#router bgp 300`	Starts the BGP routing process.
`Houston(config-router)#neighbor 192.168.7.1 remote-as 100`	Identifies a peer router at 192.168.7.1.
`Houston(config-router)#neighbor 192.168.7.1 filter-list 5 weight 2000`	Assigns a **weight** attribute of 2000 to updates from the neighbor at 192.168.7.1 that are permitted by access list 5. Access list 5 is defined in the **ip as-path access-list 5** command listed below in global configuration mode. Filter list 5 refers to the **ip as-path access-list 5** command that defines which path will be used to have this weight value assigned to it.
`Houston(config-router)#neighbor 192.168.219.1 remote-as 200`	Identifies a peer router at 192.168.219.1.
`Houston(config-router)#neighbor 192.168.219.1 filter-list 6 weight 1000`	Assigns a **weight** attribute of 1000 to updates from the neighbor at 192.168.219.1 that are permitted by access list 6. Access list 6 is defined in the **ip as-path access-list 5** command listed below in global configuration mode.
`Houston(config-router)#exit`	Returns to global configuration mode.
`Houston(config)#ip as-path access-list 5 permit _100_`	Permits updates whose AS_PATH attribute shows the update passing through autonomous system 100.
	NOTE: The _ symbol is used to form regular expressions. See the section "Regular Expressions" in this chapter (after the sections on the different attributes) for more examples.
`Houston(config)#ip as-path access-list 6 permit _200_`	Permits updates whose AS_PATH attribute shows the update passing through autonomous system 200.

The result of this configuration will have Houston forward traffic for the 172.16.10.0 network through autonomous system 100, because it has a higher weight attribute set as compared to the weight attribute set for the same update from autonomous system 200.

Using Prefix Lists and Route Maps to Manipulate the Weight Attribute

Refer to Figure 6-11 for the configuration that follows, which demonstrates how to configure the weight attribute using prefix lists and route maps. The objective here is for Houston to prefer the path through Austin to reach the 172.16.10.0/24 network.

`Houston(config)#ip prefix-list AS400_ROUTES permit 172.16.10.0/24`	Creates a prefix list that matches the 172.16.10.0/24 network belonging to autonomous system 400.
`Houston(config)#route-map SETWEIGHT permit 10`	Creates a route map called SETWEIGHT. This route map will permit traffic based on the subsequent criteria. A sequence number of 10 is assigned.
`Houston(config-route-map)#match ip address prefix-list AS400_ROUTES`	Specifies the condition under which policy routing is allowed, matching the AS400_ROUTES prefix list.
`Houston(config-route-map)#set weight 200`	Assigns a weight of 200 to any route update that meets the condition of prefix list AS400_ROUTES.
`Houston(config-route-map)#route-map SETWEIGHT permit 20`	Creates the second statement for the route map named SETWEIGHT. This route map will permit traffic based on subsequent criteria. A sequence number of 20 is assigned.
`Houston(config-route-map)#set weight 100`	Assigns a weight of 100 to all other route updates/networks learned.
`Houston(config-route-map)#exit`	Returns to global configuration mode.
`Houston(config)#router bgp 300`	Starts the BGP routing process.
`Houston(config-router)#neighbor 192.168.7.1 route-map SETWEIGHT in`	Uses the route map SETWEIGHT to filter all routes learned from neighbor 192.168.7.1.

Local Preference Attribute

Local preference is a BGP attribute that provides information to routers in the autonomous system about the path that is preferred for exiting the autonomous system. A path with a higher local preference is preferred. The local preference is an attribute that is configured on a router and exchanged among routers within the same autonomous system only.

`R1(config-router)#bgp default local-preference 150`	Changes the default local preference value from 100 to 150

NOTE: The **local-preference** value can be a number between 0 and 429,496,729. Higher is preferred. If a **local-preference** value is not set, the default is 100.

NOTE: The **local-preference** attribute is local to the autonomous system; it is exchanged between iBGP peers but not advertised to eBGP peers. Use the **local-preference** attribute to force BGP routers to prefer one exit point over another.

Using AS_PATH Access Lists with Route Maps to Manipulate the Local Preference Attribute

Route maps provide more flexibility than the **bgp default local-preference** router configuration command.

Figure 6-12 shows the network topology for the configuration that follows, which demonstrates how to configure the local-preference attribute using AS_PATH access lists with route maps. The objective here is to prefer Galveston as the autonomous system 256 exit point for all networks originating in autonomous system 300.

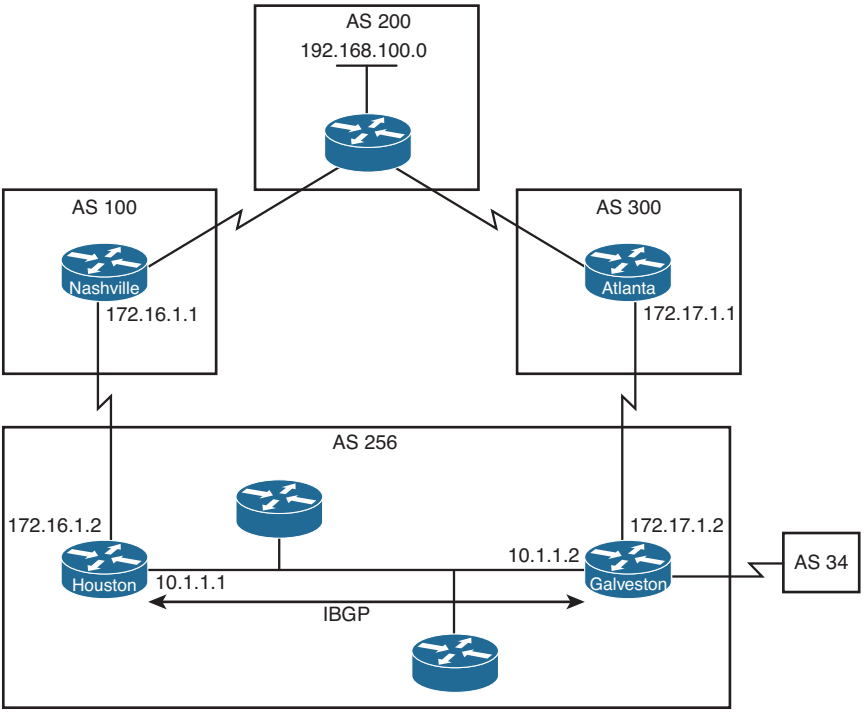


Figure 6-12 Using AS_PATH Access Lists with Route Maps to Manipulate the Local Preference Attribute

`Galveston(config)#router bgp 256`	Starts the BGP routing process.
`Galveston(config-router)#neighbor 172.17.1.1 remote-as 300`	Identifies a peer router at 172.17.1.1.
`Galveston(config-router)#neighbor 172.17.1.1 route-map SETLOCAL in`	Refers to a route map called SETLOCAL. All network update received from neighbor 172.17.1.1 will be processed by the route map.

`Galveston(config-router)#neighbor 10.1.1.1 remote-as 256`	Identifies a peer router at 10.1.1.1.
`Galveston(config-router)#exit`	Returns to global configuration mode.
`Galveston(config)#ip as-path access-list 7 permit ^300$`	Permits updates whose AS_PATH attribute starts with 300 (represented by the ^) and ends with 300 (represented by the $).
`Galveston(config)#route-map SETLOCAL permit 10`	Creates a route map called SETLOCAL. This route map will permit traffic based on subsequent criteria. A sequence number of 10 is assigned.
`Galveston(config-route-map)#match as-path 7`	Specifies the condition under which policy routing is allowed, matching the BGP ACL 7.
`Galveston(config-route-map)#set local-preference 200`	Assigns a local preference of 200 to any update originating from autonomous system 300, as defined by ACL 7.
`Galveston(config-route-map)#route-map SETLOCAL permit 20`	Creates the second statement of the route map SETLOCAL. This instance will accept all other routes.
	NOTE: Forgetting a **permit** statement at the end of the route map is a common mistake that prevents the router from learning any other routes.

AS_PATH Attribute Prepending

Autonomous system paths can be manipulated by prepending autonomous system numbers to the existing autonomous system paths. Assuming that the values of all other attributes are the same, routers will pick the shortest AS_PATH attribute; therefore, prepending numbers to the path will manipulate the decision as to the best path. Normally, AS_PATH prepending is performed on outgoing eBGP updates over the undesired return path.

Refer to Figure 6-13 for the configuration that follows, which demonstrates the commands necessary to configure the **as-path prepend** option. Assume that all basic configurations are accurate.

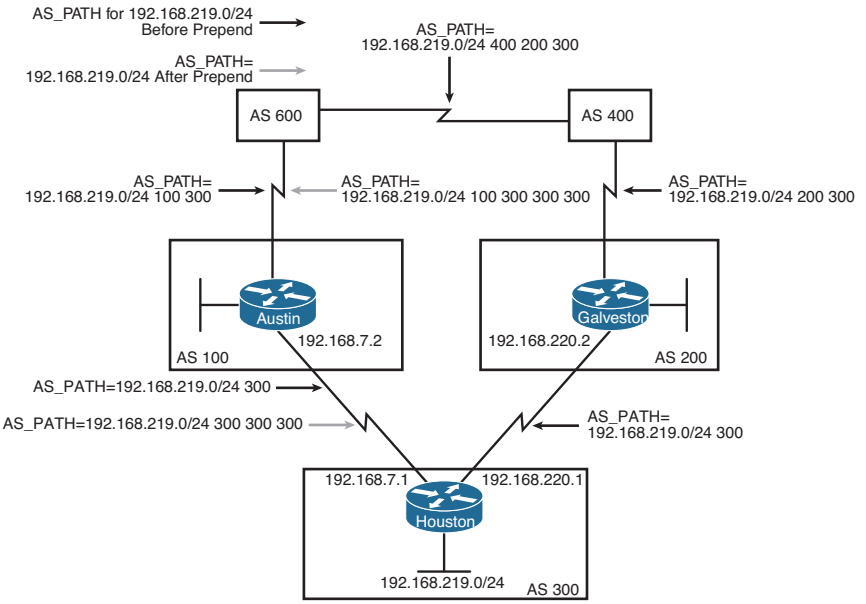


Figure 6-13 AS_PATH Attribute Prepending

In this scenario, you want to use the configuration of Houston to influence the choice of paths in autonomous system 600. Currently, the routers in autonomous system 600 have reachability information to the 192.168.219.0/24 network via two routes: via autonomous system 100 with an AS_PATH attribute of (100, 300), and via autonomous system 400 with an AS_PATH attribute of (400, 200, 300). Assuming that the values of all other attributes are the same, the routers in autonomous system 600 will pick the shortest AS_PATH attribute: the route through autonomous system 100. You will prepend, or add, extra autonomous system numbers to the AS_PATH attribute for routes that Houston advertises to autonomous system 100 to have autonomous system 600 select autonomous system 400 as the preferred path of reaching the 192.168.219.0/24 network.

`Houston(config)#router bgp 300`	Starts the BGP routing process.
`Houston(config-router)#network 192.168.219.0`	Tells the BGP process what locally learned networks to advertise.
`Houston(config-router)#neighbor 192.168.220.2 remote-as 200`	Identifies a peer router at 192.168.220.2.
`Houston(config-router)#neighbor 192.168.7.2 remote-as 100`	Identifies a peer router at 192.168.7.2.
`Houston(config-router)#neighbor 192.168.7.2 route-map SETPATH out`	Read this command to say, "All routes destined for neighbor 192.168.7.2 will have to follow the conditions laid out by the SETPATH route map."
`Houston(config-router)#exit`	Returns to global configuration mode.

`Houston(config)#route-map SETPATH permit 10`	Creates a route map named SETPATH. This route map will permit traffic based on subsequent criteria. A sequence number of 10 is assigned.
`Houston(config-route-map)#set as-path prepend 300 300`	Read this command to say, "The local router will add (prepend) the autonomous system number 300 twice to the AS_PATH attribute before sending it out to its neighbor at 192.168.7.2."

The result of this configuration is that the AS_PATH attribute of updates for network 192.168.219.0 that autonomous system 600 receives via autonomous system 100 will be (100, 300, 300, 300), which is longer than the value of the AS_PATH attribute of updates for network 192.168.219.0 that autonomous system 600 receives via autonomous system 400 (400, 200, 300).

Autonomous system 600 will choose autonomous system 400 (400, 200, 300) as the better path. This is because BGP is a path vector routing protocol that chooses the path with the least number of autonomous systems that it has to cross.

AS_PATH: Removing Private Autonomous Systems

Private autonomous system numbers (64,512 to 65,535) cannot be passed on to the Internet because they are not unique. Cisco has implemented a feature, **remove-private-as**, to strip private autonomous system numbers out of the AS_PATH list before the routes get propagated to the Internet.

Figure 6-14 shows the network for the example below which demonstrates the **remove-private-as** option. Assume that all basic configurations are accurate.

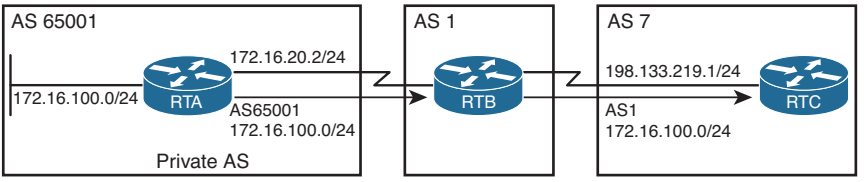


Figure 6-14 AS_PATH: Removing Private Autonomous Systems

`RTB(config)#router bgp 1`	Starts the BGP routing process.
`RTB(config-router)#neighbor 172.16.20.2 remote-as 65001`	Identifies a peer router at 172.16.20.2.
`RTB(config-router)#neighbor 198.133.219.1 remote-as 7`	Identifies a peer router at 198.133.219.1.
`RTB(config-router)#neighbor 198.133.219.1 remove-private-as`	Removes private autonomous system numbers from the path in outbound routing updates.
	NOTE: The **remove-private-as** command is available for eBGP neighbors only.

MED Attribute

The MED attribute, also called the BGP metric, can be used to indicate to eBGP neighbors what the preferred path is into an autonomous system. Unlike local preference, the MED is exchanged between autonomous systems. The MED is sent to eBGP peers. By default, a router compares the MED attribute only for paths from neighbors in the same autonomous system. The **metric** command is used to configure the MED attribute.

Figure 6-15 shows the commands necessary to configure the MED attribute. Assume that all basic configurations are accurate. The objective here is to influence Mazatlan to choose Houston as the entry point for autonomous system 300 to reach network 192.168.100.0.

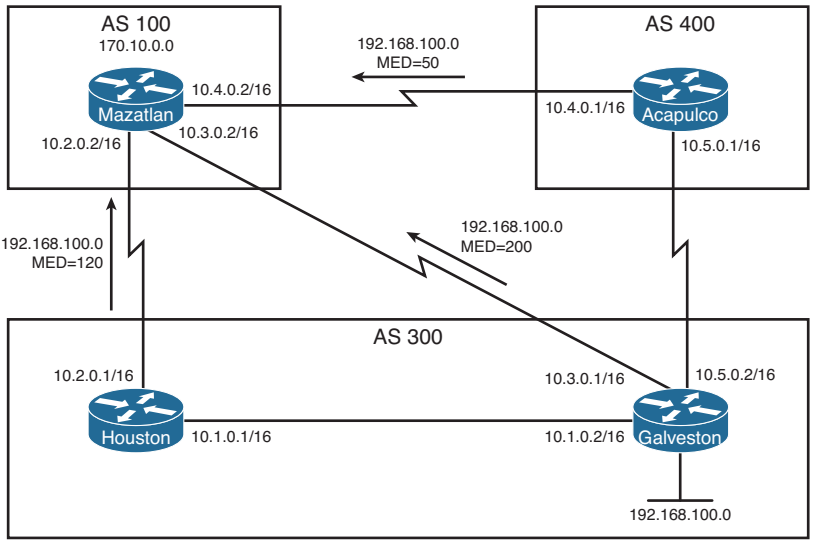


Figure 6-15 MED Attribute

`Mazatlan(config)#router bgp 100`	Starts the BGP routing process.
`Mazatlan(config-router)#neighbor 10.2.0.1 remote-as 300`	Identifies a peer router at 10.2.0.1.
`Mazatlan(config-router)#neighbor 10.3.0.1 remote-as 300`	Identifies a peer router at 10.3.0.1.
`Mazatlan(config-router)#neighbor 10.4.0.1 remote-as 400`	Identifies a peer router at 10.4.0.1.
`Acapulco(config)#router bgp 400`	Starts the BGP routing process.
`Acapulco(config-router)#neighbor 10.4.0.2 remote-as 100`	Identifies a peer router at 10.4.0.2.
`Acapulco(config-router)#neighbor 10.4.0.2 route-map SETMEDOUT out`	Refers to a route map named SETMEDOUT.

`Acapulco(config-router)#neighbor 10.5.0.2 remote-as 300`	Identifies a peer router at 10.5.0.2.
`Acapulco(config-router)#exit`	Returns to global configuration mode.
`Acapulco(config)#route-map SETMEDOUT permit 10`	Creates a route map named SETMEDOUT. This route map will permit traffic based on subsequent criteria. A sequence number of 10 is assigned.
`Acapulco(config-route-map)#set metric 50`	Sets the metric value for BGP.
`Houston(config)#router bgp 300`	Starts the BGP routing process.
`Houston(config-router)#neighbor 10.2.0.2 remote-as 100`	Identifies a peer router at 10.2.0.1.
`Houston(config-router)#neighbor 10.2.0.2 route-map SETMEDOUT out`	Refers to a route map named SETMEDOUT.
`Houston(config-router)#neighbor 10.1.0.2 remote-as 300`	Identifies a peer router at 10.1.0.2.
`Houston(config-router)#exit`	Returns to global configuration mode.
`Houston(config)#route-map SETMEDOUT permit 10`	Creates a route map named SETMEDOUT. This route map will permit traffic based on subsequent criteria. A sequence number of 10 is assigned.
`Houston(config-route-map)#set metric 120`	Sets the metric value for BGP.
`Galveston(config)#router bgp 300`	Starts the BGP routing process.
`Galveston(config-router)#neighbor 10.3.0.2 remote-as 100`	Identifies a peer router at 10.3.0.2.
`Galveston(config-router)#neighbor 10.3.0.2 route-map SETMEDOUT out`	Refers to a route map named SETMEDOUT.
`Galveston(config-router)#neighbor 10.1.0.1 remote-as 300`	Identifies a peer router at 10.1.0.1.
`Galveston(config-router)#neighbor 10.5.0.1 remote-as 400`	Identifies a peer router at 10.5.0.1
`Galveston(config-router)#exit`	Returns to global configuration mode.
`Galveston(config)#route-map SETMEDOUT permit 10`	Creates a route map named SETMEDOUT. This route map will permit traffic based on subsequent criteria. A sequence number of 10 is assigned.
`Galveston(config-route-map)#set metric 200`	Sets the metric value for BGP.

- A lower MED value is preferred over a higher MED value. The default value of the MED is 0. It is possible to change the default value of the MED using the **default-metric** command under the BGP process.

- Unlike local preference, the MED attribute is exchanged between autonomous systems, but a MED attribute that comes into an autonomous system does not leave the autonomous system.
- Unless otherwise specified, the router compares MED attributes for paths from external neighbors that are in the same autonomous system.
- If you want MED attributes from neighbors in other autonomous systems to be compared, you must configure the **bgp always-compare-med** command.

NOTE: By default, BGP compares the MED attributes of routes coming from neighbors in the same external autonomous system (such as autonomous system 300). Mazatlan can only compare the MED attribute coming from Houston (120) to the MED attribute coming from Galveston (200) even though the update coming from Acapulco has the lowest MED value. Mazatlan will choose Houston as the best path for reaching network 192.168.100.0.

To force Mazatlan to include updates for network 192.168.100.0 from Acapulco in the comparison, use the **bgp always-compare-med** router configuration command on Mazatlan:

```
Mazatlan(config)#router bgp 100
Mazatlan(config-router)#neighbor 10.2.0.1 remote-as 300
Mazatlan(config-router)#neighbor 10.3.0.1 remote-as 300
Mazatlan(config-router)#neighbor 10.4.0.1 remote-as 400
Mazatlan(config-router)#bgp always-compare-med
```

Assuming that all other attributes are the same, Mazatlan will choose Acapulco as the best next hop for reaching network 192.168.100.0.

NOTE: The most recent IETF decision about BGP MED assigns a value of infinity to the missing MED, making the route that is lacking the MED variable the least preferred. The default behavior of BGP routers that are running Cisco IOS Software is to treat routes without the MED attribute as having a MED of 0, making the route that is lacking the MED variable the most preferred. To configure the router to conform to the IETF standard, use the **bgp bestpath missing-as-worst** command.

Route Aggregation

`R1(config-router)#aggregate-address 172.16.0.0 255.255.0.0`	Creates an aggregate entry in the BGP routing table if any more-specific BGP routes are available that fall within the specified range. The aggregate route will be advertised as coming from your autonomous system and will have the atomic aggregate attribute set. More specific routes will also be advertised unless the **summary-only** option is enabled.

`R1(config-router)#aggregate-address 172.16.0.0 255.255.0.0 summary-only`	Creates the aggregate route but also suppresses advertisements of more-specific routes to all neighbors. Specific AS_PATH information to the individual subnets that fall within the summary is lost.
`R1(config-router)#aggregate-address 172.16.0.0 255.255.0.0 as-set`	Creates an aggregate entry but the path advertised for this route will be an AS_SET or list of AS_PATHs from where the individual subnets originated.

Route Reflectors

By default, a router that receives an eBGP route advertises it to its eBGP and iBGP peers. However, if it receives it through iBGP, it does not advertise it to its iBGP peers, as a loop-prevention mechanism (split horizon). Because of this behavior, the only way for all iBGP routers to receive a route after it is originated into the autonomous system is to have a full mesh of iBGP peers. This can get complex with a large number of peers. A route reflector allows a topology to get around the iBGP limitation of having to have a full mesh.

Figure 6-16 shows the commands necessary to configure BGP route reflectors. Assume that basic BGP configurations are accurate. The objective is to allow R2 to advertise to R1 the 209.165.201.0/27 network learned from R3. Without these commands, R1 will never learn the 209.165.201.0/27 network unless a full-mesh iBGP topology is built.

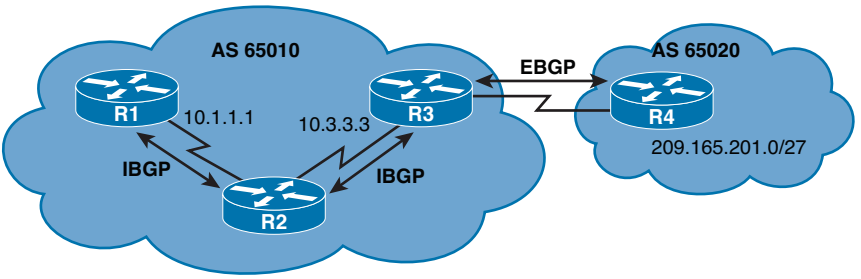


Figure 6-16 Route Reflectors

`R2(config)#router bgp 65010`	Enters BGP routing configuration mode
`R2(config-router)#neighbor 10.1.1.1 route-reflector-client`	Configures the local router as a BGP route reflector and the specified neighbor as a client
`R2(config-router)#neighbor 10.3.3.3 route-reflector-client`	Configures the local router as a BGP route reflector and the specified neighbor as a client

Regular Expressions

A *regular expression* is a pattern to match against an input string, such as those listed in the following table.

Character	Description
^	Matches the beginning of the input string
$	Matches the end of the input string
_	Matches a space, comma, left brace, right brace, the beginning of an input string, or the ending of an input stream
.	Matches any single character
*	Matches 0 or more single- or multiple-character patterns

For example, in the case of the **ip as-path access-list** command, the input string is the AS_PATH attribute.

`Router(config)#ip as-path access-list 1 permit 2150`	Will match any AS_PATH that includes the pattern of 2150.
`Router#show ip bgp regexp 2150`	Will match any AS_PATH that includes the pattern of 2150.
	NOTE: In both of these commands, not only will autonomous system 2150 be a match, but so will autonomous system 12150 or 21507.
`Router(config)#ip as-path access-list 6 deny ^200$`	Denies updates whose AS_PATH attribute starts with 200 (represented by the ^) and ends with 200 (represented by the $).
`Router(config)#ip as-path access-list 1 permit .*`	Permits updates whose AS_PATH attribute starts with any character—represented by the period (.) symbol, and repeats that character—the asterisk (*) symbol means a repetition of that character.
	NOTE: The argument of .* will match any value of the AS_PATH attribute.

Regular Expressions: Examples

Refer to the following **show ip bgp** output to see how different examples of regular expressions can help filter specific patterns:

```
R1#show ip bgp
Network          Next Hop      Metric LocPrf  Weight Path
* i172.16.0.0    172.20.50.1             100       0 65005 65004 65003 i
*>i              192.168.28.1            100       0 65002 65003 i
*>i172.24.0.0    172.20.50.1             100       0 65005 i
```

```
* i                192.168.28.1         100     0 65002 65003 65004 65005 i
*>i172.30.0.0      172.20.50.1          100     0 65005 65004 i
* i                192.168.28.1         100     0 65002 65003 65004i
*>i192.168.3.3/32 0.0.0.0                       0   32768 i
```

To find all subnets originating from autonomous system 65004 (AS_PATH ends with 65004):

```
R1#show ip bgp regexp _65004$
Network           Next Hop      Metric LocPrf  Weight Path
*>i172.30.0.0     172.20.50.1             100       0 65005 65004 i
* i               192.168.28.1            100       0 65002 65003 65004i
```

To find all subnets reachable via autonomous system 65002 (AS_PATH begins with 65002):

```
R1#show ip bgp regexp ^65002_
Network        Next Hop    Metric LocPrf Weight Path
*>i172.16.0.0  192.168.28.1          100      0 65002 65003 i
* i172.24.0.0  192.168.28.1          100      0 65002 65003 65004 65005 i
* i172.30.0.0  192.168.28.1          100      0 65002 65003 65004i
```

To find all routes transiting through autonomous system 65005:

```
R1#show ip bgp regexp _65005_
Network       Next Hop    Metric LocPrf Weight Path
* i172.16.0.0 172.20.50.1           100      0 65005 65004 65003 i
*>i172.24.0.0 172.20.50.1           100      0 65005 i
* i           192.168.28.1          100      0 65002 65003 65004 65005 i
*>i172.30.0.0 172.20.50.1           100      0 65005 65004 i
```

To find subnets that originate from R1's autonomous system (AS_PATH is blank):

```
R1#show ip bgp regexp ^$
Network             Next Hop      Metric LocPrf  Weight Path
*>i192.168.3.3/32   0.0.0.0          0   32768   i
```

BGP Route Filtering Using Access Lists and Distribute Lists

Figure 6-17 shows the commands necessary to configure route filters using access lists and distribute lists.

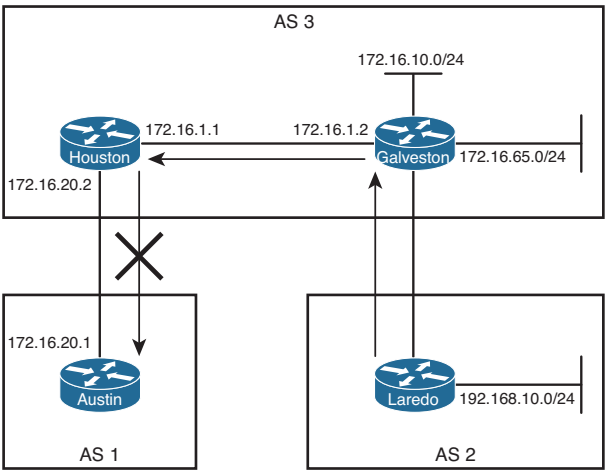


Figure 6-17 BGP Route Filtering Using Access Lists and Distribute Lists

In this scenario, we want to have Houston filter updates to Austin so that it does not include the 192.168.10.0/24 network.

`Houston(config)#router bgp 3`	Starts the BGP routing process
`Houston(config-router)#neighbor 172.16.1.2 remote-as 3`	Identifies a peer router at 172.16.1.2
`Houston(config-router)#neighbor 172.16.20.1 remote-as 1`	Identifies a peer router at 172.16.20.1
`Houston(config-router)#neighbor 172.16.20.1 distribute-list 1 out`	Applies a filter of ACL 1 to updates sent to neighbor 172.16.20.1
`Houston(config-router)#exit`	Returns to global configuration mode
`Houston(config)#access-list 1 deny 192.168.10.0 0.0.0.255`	Creates the filter to prevent the 192.168.10.0/24 network from being part of the routing update
`Houston(config)#access-list 1 permit any`	Creates the filter that allows all other networks to be part of the routing update

TIP: A standard ACL offers limited functionality. If you want to advertise the aggregate address of 172.16.0.0/16 but not the individual subnet, a standard ACL will not work. You need to use an extended ACL.

When you are using extended ACLS with BGP route filters, the extended ACL will first match the network address and *then* match the subnet mask of the prefix. To do this, both the network and the netmask are paired with their own wildcard bitmask:

```
Router(config)#access-list 101 permit ip 172.16.0.0 0.0.255.255
  255.255.0.0 0.0.0.0
```

To help overcome the confusing nature of this syntax, Cisco IOS Software introduced the **ip prefix-list** command in Cisco IOS Release 12.0.

Configuration Example: Using Prefix Lists and AS_PATH Access Lists

Figure 6-18 shows the network topology for the configuration that follows, which demonstrates how to configure prefix lists and AS_PATH access lists. Assume that all BGP and basic configurations are accurate. There are two objectives here. The first is to allow CE1 and CE2 to only learn ISP routes with a mask greater than /15 (ge 16) and less than /25 (le 24). The second is to ensure that autonomous system 65000 does not become a transit autonomous system for ISP1 to reach ISP2 (and vice versa).

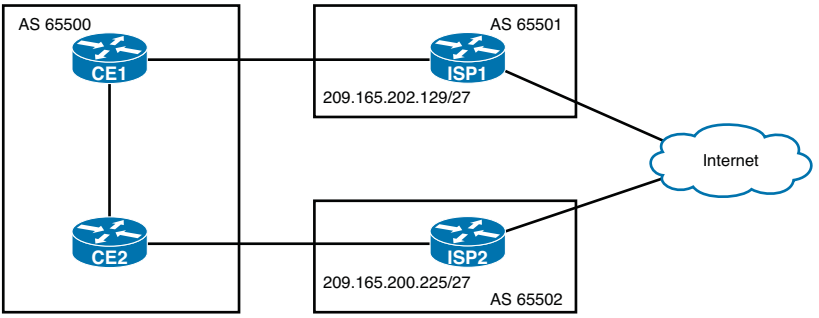


Figure 6-18 Configuration Example: Using Prefix Lists and AS_PATH Access Lists

`CE1(config)#ip prefix-list ISP1 permit 0.0.0.0 ge 16 le 24`	Creates a prefix list which only permits routes with a mask between 16 and 24
`CE1(config)#ip as-path access-list 1 permit ^$`	Creates an AS_PATH access list matching routes that only originate from within autonomous system 65500
`CE1(config)#router bgp 65000`	Starts the BGP routing process
`CE1(config-router)#neighbor 209.165.202.129 prefix-list ISP1 in`	Assigns the ISP1 prefix list to neighbor 209.165.202.129 (ISP1) for all routes learned from that neighbor
`CE1(config-router)#neighbor 209.165.202.129 filter-list 1 out`	Assigns the AS_PATH access list to neighbor 209.165.202.129 (ISP1) for all routes sent to that neighbor
`CE2(config)#ip prefix-list ISP2 permit 0.0.0.0 ge 16 le 24`	Creates a prefix list that only permits routes with a mask between 16 and 24
`CE2(config)#ip as-path access-list 1 permit ^$`	Creates an AS_PATH access list matching routes that only originate from within autonomous system 65500
`CE2(config)#router bgp 65000`	Starts the BGP routing process

`CE2(config-router)#neighbor 209.165.200.225 prefix-list ISP2 in`	Assigns the ISP2 prefix list to neighbor 209.165.200.225 (ISP2) for all routes learnt from that neighbor
`CE2(config-router)#neighbor 209.165.200.225 filter-list 1 out`	Assigns the AS_PATH access list to neighbor 209.165.200.225 (ISP2) for all routes sent to that neighbor

BGP Peer Groups

To ease the burden of configuring a large number of neighbors with identical or similar parameters (for example, route maps, filter lists, or prefix lists), the concept of peer groups was introduced. The administrator configures the peer group with all the BGP parameters that are to be applied to many BGP peers. Actual BGP neighbors are bound to the peer group, and the network administrator applies the peer group configuration on each of the BGP sessions.

Figure 6-19 shows the network topology for the configuration that follows, which demonstrates how to configure peer groups. Assume that all BGP, OSPF, and basic configurations are accurate.

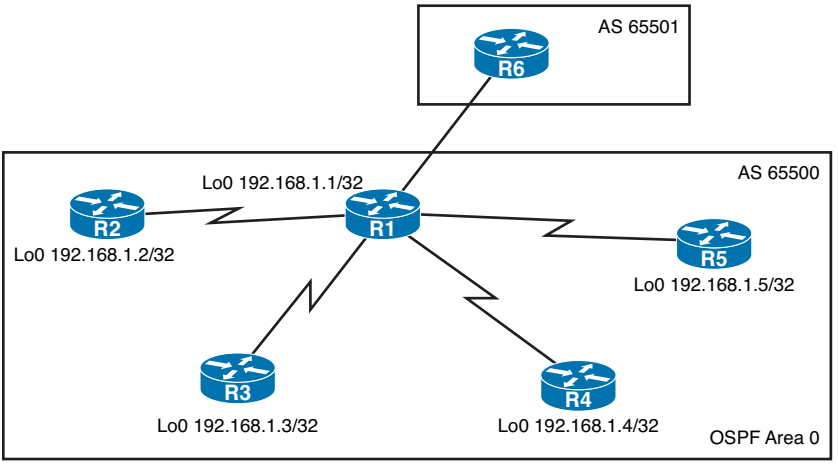


Figure 6-19 BGP Peer Groups

`R1(config)#router bgp 65500`	Starts the BGP routing process
`R1(config-router)#neighbor INTERNAL peer-group`	Creates a BGP peer group called INTERNAL
`R1(config-router)#neighbor INTERNAL remote-as 65500`	Assigns a first parameter to the peer group
`R1(config-router)#neighbor INTERNAL next-hop-self`	Assigns a second parameter to the peer group

`R1(config-router)#neighbor INTERNAL update-source loopback0`	Assigns a third parameter to the peer group
`R1(config-router)#neighbor INTERNAL route-reflector-client`	Assigns a fourth parameter to the peer group
`R1(config-router)#neighbor 192.168.1.2 peer-group INTERNAL`	Assigns the peer group to neighbor R2
`R1(config-router)#neighbor 192.168.1.3 peer-group INTERNAL`	Assigns the peer group to neighbor R3
`R1(config-router)#neighbor 192.168.1.4 peer-group INTERNAL`	Assigns the peer group to neighbor R4
`R1(config-router)#neighbor 192.168.1.5 peer-group INTERNAL`	Assigns the peer group to neighbor R5

The result here is that all four iBGP neighbors have the same basic BGP configuration assigned to them.

TIP: A peer group can be, among others, configured to do the following:

- Use the IP address of a specific interface as the source address when opening the TCP session or use the next-hop-self feature.
- Use, or not use, the eBGP multihop function.
- Use, or not use, MD5 authentication on the BGP sessions.
- Filter out any incoming or outgoing routes using a prefix list, a filter list, and a route map.
- Assign a particular weight value to the routes that are received.

MP-BGP

Original BGP was designed to carry only IPv4 specific information. A recent extension was defined to also support other protocols like IPv6. This extension is called MP-BGP (Multiprotocol BGP). You can run MP-BGP over IPv4 or IPv6 transport and can exchange routes for IPv4, IPv6, or both. BGP uses TCP for peering, and this has no relevance to the routes carried inside the BGP exchanges. Both IPv4 and IPv6 can be used to transport a TCP connection on the network layer.

Configure MP-BGP Using Address Families to Exchange IPv4 and IPv6 Routes

In this example, MP-BGP is used to exchange both IPv4 and IPv6 routes. The IPv4 routes will use an IPv4 TCP connection, and the IPv6 routes will use an IPv6 TCP connection.

Figure 6-20 shows the network topology for the configuration that follows, which demonstrates how to configure MP-BGP using address families to exchange both IPv4 and IPv6 routes. Assume that all basic configurations are accurate.

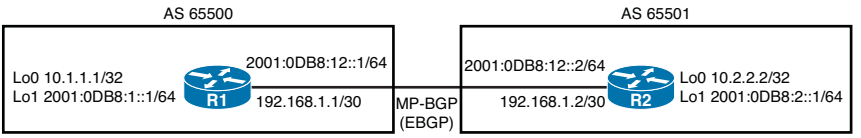


Figure 6-20 Configuring MP-BGP Using Address Families to Exchange IPv4 and IPv6 Routes

`R1(config)#ipv6 unicast-routing`	Enables the forwarding of IPV6 unicast datagrams globally on the router.
`R1(config)#router bgp 65500`	Starts the BGP routing process.
`R1(config-router)#neighbor 2001:0DB8:12::2 remote-as 65501`	Configures R2 as an IPv6 BGP neighbor.
`R1(config-router)#neighbor 192.168.1.2 remote-as 65501`	Configures R2 as an IPv4 BGP neighbor.
`R1(config-router)#address-family ipv4 unicast`	Enters IPv4 address family configuration mode for unicast address prefixes.
	TIP: Unicast address prefixes are the default when IPv4 address prefixes are configured.
`R1(config-router-af)#neighbor 192.168.1.2 activate`	Enables the exchange of IPv4 BGP information with R2. The IPv4 neighbors will be automatically activated, so this command is optional.
`R1(config-router-af)#network 10.1.1.1 mask 255.255.255.255`	Advertises an IPv4 network into BGP.
`R1(config-router-af)#exit`	Exits the IPv4 address family configuration mode.
`R1(config-router)#address-family ipv6 unicast`	Enters IPv6 address family configuration mode for unicast address prefixes.
	TIP: Unicast address prefixes are the default when IPv6 address prefixes are configured.
`R1(config-router-af)#neighbor 2001:0DB8:12::2 activate`	Enables the exchange of IPv6 BGP information with R2.
`R1(config-router-af)#network 2001:0DB8:1::1/64`	Advertises an IPv6 network into BGP.
`R2(config)#ipv6 unicast-routing`	Enables the forwarding of IPv6 unicast datagrams globally on the router.
`R2(config)#router bgp 65501`	Starts the BGP routing process.
`R2(config-router)#neighbor 2001:0DB8:12::1 remote-as 65500`	Configures R1 as an IPv6 BGP neighbor.
`R2(config-router)#neighbor 192.168.1.1 remote-as 65500`	Configures R1 as an IPv4 BGP neighbor.

`R2(config-router)#address-family ipv4 unicast`	Enters IPv4 address family configuration mode for unicast address prefixes.
	TIP: Unicast address prefixes are the default when IPv4 address prefixes are configured.
`R2(config-router-af)#neighbor 192.168.1.1 activate`	Enables the exchange of IPv4 BGP information with R1. The IPv4 neighbors will be automatically activated, so this command is optional.
`R2(config-router-af)#network 10.2.2.2 mask 255.255.255.255`	Advertises an IPv4 network into BGP.
`R2(config-router-af)#exit`	Exits the IPv4 address family configuration mode.
`R2(config-router)#address-family ipv6 unicast`	Enters IPv6 address family configuration mode for unicast address prefixes.
	TIP: Unicast address prefixes are the default when IPv6 address prefixes are configured.
`R2(config-router-af)#neighbor 2001:0DB8:12::1 activate`	Enables the exchange of IPv6 BGP information with R1.
`R2(config-router-af)#network 2001:0DB8:2::1/64`	Advertises an IPv6 network into BGP.

NOTE: By default, BGP sets its router ID to the IPv4 address of the highest address of the loopback interface, or if no loopback exists, to the highest IP address of the physical interface. If the router running BGP over IPv6 transport has no IPv4 interfaces configured, you need to manually specify the BGP router ID using the **bgp router-id** *IPv4_address* BGP configuration command.

Verifying MP-BGP

`Router#show bgp ipv6 unicast`	Provides output similar to the **show ip bgp** command, except it is IPv6 specific
`Router#show bgp ipv6 unicast summary`	Provides output similar to the **show ip bgp summary** command, except it is IPv6 specific
`Router#show bgp ipv6 unicast neighbors`	Provides output similar to the **show ip bgp neighbors** command, except it is IPv6 specific.
`Router#show ipv6 route bgp`	Displays the content of the IPv6 routing table

This page intentionally left blank

CHAPTER 7

Routers and Routing Protocol Hardening

This chapter provides information about the following topics:

- Securing Cisco routers according to recommended practices
  - Securing Cisco IOS routers checklist
  - Components of a router security policy
  - Configuring passwords
  - Password encryption
  - Configuring SSH
    - Verifying SSH
  - Restricting virtual terminal access
  - Securing Access to the infrastructure using router ACLs
  - Configuring secure SNMP
    - Securing SNMPv1 or SNMPv2
    - Securing SNMPv3
    - Verifying SNMP
  - Configuration backups
  - Implementing logging
    - Configuring syslog
    - Syslog message formats
    - Syslog severity levels
    - Syslog message example
    - Configuring NetFlow
    - Verifying NetFlow
  - Disabling unused services
- Configuring Network Time Protocol
  - NTP configuration
  - NTP design
  - Securing NTP
  - Verifying and troubleshooting NTP
  - SNTP

- Setting the clock on a router
- Using time stamps
- Configuration example: NTP

- Authentication of routing protocols
  - Authentication options for different routing protocols
  - Authentication for EIGRP
    - Configuring EIGRP authentication
    - Configuring authentication in named EIGRP
    - Verifying and troubleshooting EIGRP authentication
  - Authentication for OSPF
    - Configuring OSPFv2 authentication: simple
    - Configuring OSPFv2 authentication: using MD5 encryption
    - Configuring OSPFv2 authentication: using SHA encryption
    - Configuring OSPFv3 authentication and encryption
    - Verifying OSPFv2 and OSPFv3 authentication
  - Authentication for BGP and BGP for IPv6
    - Configuring authentication between BGP peers
    - Verifying BGP and BGP for IPv6 authentication

Securing Cisco Routers According to Recommended Practices

Router security is critical to network security. A compromised router can cause the network to be compromised on a larger scale. The following sections deal with different ways to help secure your Cisco IOS routers.

Securing Cisco IOS Routers Checklist

Table 7-1 shows the checklist that you should use when securing Cisco IOS routers.

TABLE 7-1 Securing Cisco IOS Routers Checklist

Recommended Practice	Y/N
Set up and follow security policy	
Use encrypted passwords	
Secure access to the router using access control lists (ACLs)	
Use secure management protocols	
Periodically back up configurations	
Implement logging	
Disable unused services	

Components of a Router Security Policy

Table 7-2 shows the items that should be part of any router security policy.

TABLE 7-2 Router Security Policy

Password encryption and complexity settings
Authentication settings
Management access settings
Unneeded services settings
Ingress/egress filtering settings
Routing protocol security settings
Configuration maintenance
Change management
Router redundancy
Monitoring and incident handling
Security updates

Configuring Passwords

These commands work on both routers and switches.

`Edmonton(config)#enable password cisco`	Sets the enable password. This password is stored as clear text.
`Edmonton(config)#enable secret class`	Sets enable secret password. This password is stored using a cryptographic has function (SHA-256).
`Edmonton(config)#line console 0`	Enters console line mode.
`Edmonton(config-line)#password console`	Sets console line mode password to console.
`Edmonton(config-line)#login`	Enables password checking at login.
`Edmonton(config)#line vty 0 4`	Enters vty line mode for all five vty lines.
`Edmonton(config-line)#password telnet`	Sets vty password to telnet.
`Edmonton(config-line)#login`	Enables password checking at login.
`Edmonton(config)#line aux 0`	Enters auxiliary line mode.
`Edmonton(config-line)#password backdoor`	Sets auxiliary line mode password to backdoor.
`Edmonton(config-line)#login`	Enables password checking at login.

CAUTION: The **enable secret** *password* is encrypted by default using the SHA-256 cryptographic hash function. The **enable password** is not; it is stored as clear text. For this reason, recommended practice is that you *never* use the **enable password** command. Use only the **enable secret** *password* command in a router or switch configuration.

TIP: You can set both **enable secret** *password* and **enable** *password* to the same password. However, doing so defeats the use of encryption.

CAUTION: Line passwords are stored as clear text. They should be encrypted using the **service password-encryption** command as a bare minimum. However, this encryption method is weak and easily reversible. It is therefore recommended to enable authentication by user local names and passwords. Local names and passwords can be stored as SHA-256 encrypted passwords.

TIP: The best place to store passwords is an external AAA (authentication, authorization, and accounting) server.

Password Encryption

`Edmonton(config)#service password-encryption`	Applies a Vigenere cipher (type 7) weak encryption to passwords
`Edmonton(config)#enable password cisco`	Sets the enable password to cisco
`Edmonton(config)#line console 0`	Moves to console line mode
`Edmonton(config-line)#password cisco`	Continue setting passwords as above
	...
`Edmonton(config)#no service password-encryption`	Turns off password encryption

CAUTION: If you have turned on service password encryption, used it, and then turned it off, any passwords that you have encrypted will stay encrypted. New passwords will remain unencrypted.

TIP: If you want to enter in a password that is already encrypted with the SHA-256 hash (for example, if you are copying an existing configuration into the router), you have to instruct the router that the password is already encrypted. To do this, use the **enable secret 4** command:

```
Edmonton(config)#enable secret 4 Rv4kArhts7yA2xd8BD2YTVbts
```

To specify the message digest 5 (MD5) authentication hash of the password, use the **enable secret 5** command, followed by the MD5 hash of the password:

```
Edmonton(config)#enable secret 5 00271A5307542A02D22842
```

TIP: The **service password-encryption** command will work on the following passwords:

Username

Authentication key

Privileged command

Console

Virtual terminal line access

BGP neighbors

Passwords using this encryption are shown as type 7 passwords in the router configuration:

```
Edmonton#show running-config
                  <output omitted>
enable secret 4 Rv4kArhts7yA2xd8BD2YTVbts (4 signifies SHA-256 hash)
<output omitted>
line con 0
 password 7 00271A5307542A02D22842 (7 signifies Vigenere cipher)
line vty 0 4
 password 7 00271A5307542A02D22842 (7 signifies Vigenere cipher)
<output omitted>
R1#
```

Configuring SSH

Although Telnet is the most popular way of accessing a router, it is the most unsecure. Secure Shell (SSH) provides an encrypted alternative for accessing a router.

CAUTION: SSH Version 1 implementations have known security issues. It is recommended to use SSH Version 2 whenever possible.

NOTE: The device name cannot be the default *switch* (on a switch) or *router* (on a router). Use the **hostname** command to configure a new hostname of the device.

NOTE: The Cisco implementation of SSH requires Cisco IOS Software to support Rivest, Shamir, Adleman (RSA) authentication and minimum Data Encryption Standard (DES) encryption (a cryptographic software image).

`Edmonton(config)#username Roland password tower`	Creates a locally significant username/password combination. These are the credentials you must enter when connecting to the router with SSH client software.
`Edmonton(config)#username Roland privilege 15 secret tower`	Creates a locally significant username of Roland with privilege level 15. Assigns a secret password of tower.
`Edmonton(config)#ip domain-name test.lab`	Creates a host domain for the router.
`Edmonton(config)#crypto key generate rsa modulus 2048`	Enables the SSH server for local and remote authentication on the router and generates an RSA key pair. The number of modulus bits on the command line is 2048 bits. The size of the key modulus is 360–4096 bits.
`Edmonton(config)#ip ssh version 2`	Enables SSH version 2 on the device.

NOTE: To work, SSH requires a local username database, a local IP domain, and an RSA key to be generated.

`Edmonton(config)#line vty 0 4`	Move to vty configuration mode for all five vty lines of the router.
	NOTE: Depending on the IOS and platform, there may be more than 5 vty lines.
`Edmonton(config-line)#login local`	Enables password checking on a per-user basis. Username and password will be checked against the data entered with the **username** global configuration command.
`Edmonton(config-line)#transport input ssh`	Limits remote connectivity to ssh connections only—disables Telnet.

Verifying SSH

`Edmonton#show ip ssh`	Verifies that SSH is enabled
`Edmonton#show ssh`	Checks the SSH connection to the device

Restricting Virtual Terminal Access

`Edmonton(config)#access-list 2 permit host 172.16.10.2`	Permits host from source address of 172.16.10.2 to telnet/SSH into this router based on where this ACL is applied.
`Edmonton(config)#access-list 2 permit 172.16.20.0 0.0.0.255`	Permits anyone from the 172.16.20.*x* address range to telnet/SSH into this router based on where this ACL is applied.
	The implicit deny statement restricts anyone else from being permitted to telnet/SSH.
`Edmonton(config)#access-list 2 deny any log`	Any packets that are denied by this ACL will be logged for review at a later time. This line will be used instead of the implicit deny line.
`Edmonton(config)#line vty 0 4`	Moves to vty line configuration mode.
	NOTE: Depending on the IOS and platform, there may be more than 5 vty lines.
`Edmonton(config-line)access-class 2 in`	Applies this ACL to all vty virtual interfaces in an inbound direction.

TIP: When restricting access on vty lines, use the **access-class** command rather than the **access-group** command, which is used when applying an ACL to a physical interface.

CAUTION: Do not apply an ACL intending to restrict vty traffic on a physical interface. If you apply to a physical interface, *all* packets will be compared to the ACL before it can continue on its path to its destination. This can lead to a large reduction in router performance. An ACL on a physical interface would also have to specify the SSH or Telnet port number that you are trying to deny, in addition to identifying all of the router's addresses that you could potentially SSH/telnet to.

Securing Access to the Infrastructure Using Router ACLs

As opposed to device-centric models, infrastructure ACLs filter traffic on the network edge (that is, routers that accept IP traffic from network users or external networks).

TIP: Infrastructure ACLs are typically applied in the input direction on the interface that connects to the network users or external networks.

`Edmonton(config)#ip access-list extended ACL-INFRASTRUCTURE-IN`	Creates an extended NAMED access list and moves to named ACL configuration mode
`Edmonton(config-ext-nacl)#remark ---Deny IP Fragments---`	Creates a comment (up to 100 characters) for the ACL
`Edmonton(config-ext-nacl)#deny tcp any any fragments`	Checks for and denies any noninitial TCP fragments
`Edmonton(config-ext-nacl)#deny udp any any fragments`	Checks for and denies any noninitial UDP fragments
`Edmonton (config-ext-nacl)#deny icmp any any fragments`	Checks for and denies any noninitial ICMP fragments
`Edmonton (config-ext-nacl)#deny ip any any fragments`	Checks for and denies any noninitial IP fragments
`Edmonton(config-ext-nacl)#remark ---Permit required connections for routing protocols and network management---`	Creates a comment (up to 100 characters) for the ACL
`Edmonton(config-ext-nacl)#permit tcp host` *`trusted-ebgp-peer`* `host` *`local-ebgp-address`* `eq 179`	Permits BGP sessions from trusted hosts to local IP addresses
`Edmonton(config-ext-nacl)#permit tcp host` *`trusted-ebgp-peer`* `eq 179 host` *`local-ebgp-address`*	Permits BGP sessions from trusted hosts to local IP addresses
`Edmonton(config-ext-nacl)#permit tcp host` *`trusted-management-stations`* `any eq 22`	Permits SSH management traffic from trusted management stations
`Edmonton(config-ext-nacl)#permit udp host` *`trusted-management-servers`* `any eq 161`	Permits SNMP management traffic from trusted management servers
`Edmonton(config-ext-nacl)#remark ---ICMP ECHO (Ping) from trusted management stations---`	Creates a comment (up to 100 characters) for the ACL

`Edmonton(config-ext-nacl)#permit icmp host` *`trusted-management-stations`* `any echo`	Permits echo (ping) traffic from trusted management stations
`Edmonton(config-ext-nacl)#remark ---Deny all other IP traffic to any network device---`	Creates a comment (up to 100 characters) for the ACL
`Edmonton(config-ext-nacl)#deny ip any` *`infrastructure-address-space`*	Denies all other traffic to any infrastructure device
`Edmonton(config-ext-nacl)#remark ---Permit transit traffic---`	Creates a comment (up to 100 characters) for the ACL
`Edmonton(config-ext-nacl)#permit ip any any`	Allows all transit traffic across the router
`Edmonton(config-ext-nacl)#exit`	Returns to global configuration mode
`Edmonton(config)#interface gigabitethernet 0/0`	Move to interface configuration mode
`Edmonton(config-if)#ip access-group ACL-INFRASTRUCTURE-IN in`	Assigns the ACL to the interface in an inbound direction

Configuring Secure SNMP

Simple Network Management Protocol (SNMP) is the most commonly used network management protocol. It is important to restrict SNMP access to the routers on which it is enabled.

TIP: If SNMP is not required on a router, you should turn it off by using the **no snmp-server** command at the global configuration mode prompt.

```
Edmonton(config)#no snmp-server
```

NOTE: Beginning with SNMPv3, methods to ensure the secure transmission of data between manager and agent were added. You can now define a security policy per group, or limit IP addresses to which its members can belong. You will now have to define encryption and hashing algorithms and passwords for each user.

Table 7-3 Shows the different SNMP security models.

TABLE 7-3 SNMP Security Models

SNMP Version	Access Mode	Authentication	Encryption
SNMPv1	noAuthNoPriv	Community string	No
SNMPv2	noAuthNoPriv	Community string	No
SNMPv3	noAuthNoPriv	Username	No
	authNoPriv	MD5 or SHA-1	No
	authPriv	MD5 or SHA-1	DES, 3DES, or AES

TIP: The SNMP security levels are as follows:

- **noAuthNoPriv:** Authenticates SNMP messages using a community string. No encryption provided.
- **authNoPriv:** Authentication SNMP messages using either HMAC with MD5 or SHA-1. No encryption provided.
- **authPriv:** Authenticates SNMP messages by using either HMAC-MD5 or SHA. Encrypts SNMP Messages using DES, 3DES, or AES.
- **priv:** Does not authenticate SNMP messages. Encrypts only DES or AES.

TIP: SNMPv3 provides all three security level options. It should be used wherever possible.

TIP: If SNMPv3 cannot be used, secure SNMPv1 or SNMPv2 by using uncommon, complex community strings and by enabling read-only access.

TIP: If community strings are also used for SNMP traps, they must be different from community strings for get and set methods. This is considered best practice.

Securing SNMPv1 or SNMPv2

`Edmonton(config)#snmp-server community C0mpl3xAdmin ro 98`	Sets a community string named C0mpl3xAdmin. It is read-only and refers to ACL 98 to limit SNMP access to the authorized hosts.
	NOTE: A named ACL can be used as well.
`Edmonton(config)#access-list 98 permit host 192.168.10.3`	Creates an ACL that will limit the SNMP access to the specific host of 192.168.10.3.
`Edmonton(config)#snmp-server host 192.168.10.3 AdminC0mpl3x`	Sets the Network Management System (NMS) IP address of 192.168.10.3 and the community string of AdminC0mpl3x, which will be used to protect the sending of the SNMP traps. The community string is also used to connect to the host.

Securing SNMPv3

`Edmonton(config)#access-list 99 permit 10.1.1.0 0.0.0.255`	Creates an ACL that will be used to limit SNMP access to the local device from SNMP managers within the 10.1.1.0/24 subnet.
`Edmonton(config)#snmp-server view MGMT SysUpTime included`	Defines an SNMP view named MGMT and an OID name of SysUpTime. This OID is included in the view.

`Edmonton(config)#snmp-server view MGMT ifDescr included`	Defines an SNMP view named MGMT and an OID name of ifDescr. This OID is included in the view.
`Edmonton(config)#snmp-server view MGMT ifAdminStatus included`	Defines an SNMP view named MGMT and an OID name of ifAdminStatus. This OID is included in the view.
`Edmonton(config)#snmp-server view MGMT ifOperStatus included`	Defines an SNMP view named MGMT and an OID name of ifOperStatus. This OID is included in the view.
`Edmonton(config)#snmp-server group groupAAA v3 priv read MGMT write MGMT access 99`	Defines SNMPv3 group. The group is configured with the following: "authPriv" security level = **groupAAA v3 priv** SNMP read and write access limited to devices defined in access list 99 = **read MGMT write MGMT access 99**
`Edmonton(config)#snmp-server user userAAA groupAAA v3 auth sha itsa5ecret priv aes 256 another5ecret`	Configures a new user to the SNMP group with authentication and encryption: User and group = **snmp-server user userAAA groupAAA** Password for authentication = **auth sha itsa5ecret** Password for encryption = **priv aes 256 another5ecret**
`Edmonton(config)#snmp-server enable traps`	Enables SNMP traps.
`Edmonton(config)#snmp-server host 10.1.1.50 traps version 3 priv userAAA cpu port-security`	Defines a receiving manager for traps at ip address 10.1.1.50. UserAAA will have authPriv security level (**priv** events limited to CPU and port security-related events) = **cpu port-security**
`Edmonton(config)#snmp-server ifindex persist`	Prevents index shuffle.
	NOTE: SNMP does not identify object instances by names but by numeric indexes. Index number may change due to instance changes, such as a new interface being configured. This command will guarantee index persistence when changes occur.

Verifying SNMP

`Edmonton#show snmp`	Provides basic information about SNMP configuration
`Edmonton#show snmp view`	Provides information about SNMP views
`Edmonton#show snmp group`	Provides information about configured SNMP groups
`Edmonton#show snmp user`	Provides information about configured SNMP users

Configuration Backups

It is very important to keep a copy of a router's configuration in a location other than NVRAM. Automated jobs can be set up to copy configurations from the router at regular intervals to local or remote file systems.

`Edmonton(config)#archive`	Enters archive configuration mode.
`Edmonton(config-archive)#path ftp://admin:cisco123@192.168.10.3/$h.cfg`	Sets the base file path for the remote location of the archived configuration. The FTP server is located at 192.168.10.3. The Username to access the FTP Server is admin. The password is cisco123. The path can be a local or a remote path. Path options include **flash**, **ftp**, **http**, **https**, **rcp**, **scp**, or **tftp**. Two variables can be used with the **path** command: **$h** will be replaced with device hostname. **$t** will be replaced with date and time of the archive. If you do not use **$t**, the names of the new files will be appended with a version number so as to differentiate from the previous configurations from the same device.
`Edmonton(config-archive)#time-period 1440`	Sets the period of time (in minutes) in which to automatically archive the running-config. This number can range from 1 to 525,600 minutes. 1440 minutes = 1 day. 525,600 minutes = 1 year.
`Edmonton(config-archive)#write-memory`	Enables automatic backup generation during write memory.
`Edmonton#show archive`	Displays the list of archives. This command will also have a pointer to the most recent archive.

TIP: To create an archive copy manually, use the **archive config** command from EXEC mode:

```
Edmonton#archive config
```

TIP: When the **write-memory** command is enabled, the **copy running-config startup-config** command will trigger an archive to occur.

Implementing Logging

It is important for network administrators to implement logging to get the insight into what is occurring in their network. When a router reloads, all local logs are lost, so it is important to implement logging to an external destination. These next sections deal with the different mechanisms that you can use to configure logging to a remote location.

Configuring Syslog

`Edmonton(config)#logging on`	Enables logging to all supported destinations.
`Edmonton(config)#logging 192.168.10.53`	Logging messages will be sent to a syslog server host at address 192.168.10.53.
`Edmonton(config)#logging sysadmin`	Logging messages will be sent to a syslog server host named sysadmin.
`Edmonton(config)#logging trap x`	Sets the syslog server logging level to value *x*, where *x* is a number between 0 and 7 or a word defining the level. Table 7-4 provides more details.
`Edmonton(config)#service sequence-numbers`	Stamps syslog messages with a sequence number.
`Edmonton(config)#service timestamps log datetime`	Syslog messages will now have a time stamp included.

Syslog Message Format

The general format of Syslog messages generated on Cisco IOS Software is as follows:

```
seq no:timestamp: %facility-severity-MNEMONIC:description
```

Item in Syslog Message	Definition
seq no	Sequence number. Stamped only if the **service sequence-numbers** global configuration command is configured.
timestamp	Date and time of the message. Appears only if the **service timestamps log datetime** global configuration command is configured.
facility	The facility to which the message refers (SNMP, SYS, and so on).

Item in Syslog Message	Definition
severity	Single-digit code from 0 to 7 that defines the severity of the message. See Table 7-4 for descriptions of the levels.
MNEMONIC	String of text that uniquely defines the message.
description	String of text that contains detailed information about the event being reported.

Syslog Severity Levels

Table 7-4 shows that there are eight levels of severity in logging messages.

TABLE 7-4 Syslog Severity Levels

Level #	Level Name	Description
0	Emergencies	System is unusable.
1	Alerts	Immediate action needed.
2	Critical	Critical conditions.
3	Errors	Error conditions.
4	Warnings	Warning conditions.
5	Notifications	Normal but significant conditions.
6	Informational	Informational messages (default level).
7	Debugging	Debugging messages.

Setting a level means you will get that level and everything numerically below it. Level 6 means you will receive messages for levels 0 through 6.

Syslog Message Example

The easiest syslog message to use as an example is the one that shows up every time you exit from global configuration back to privileged EXEC mode. You have just finished entering a command and you want to save your work, but after you type in **exit** you see something like this:

(Your output will differ depending if you have sequence numbers and/or time/date stamps configured).

```
Edmonton(config)#exit
Edmonton#
*Jun 23:22:45:20.878: %SYS-5-CONFIG_I: Configured from console by
  console
Edmonton#
```

So what does this all mean?

- No sequence number is part of this message.
- The message occurred at June 23, at 22:45:20.878 (or 10:45 PM, and 20.878 seconds!).

- It is a sys message, and it is level 5 (a notification).
- It is a config message, and specifically we are being told that the configuration occurred from the console.

Configuring NetFlow

NetFlow is an application for collecting IP traffic information. It is used for network accounting and security auditing.

CAUTION: NetFlow consumes additional memory. If you have limited memory, you might want to preset the size of the NetFlow cache to contain a smaller amount of entries. The default cache size depends on the platform of the device.

`Edmonton(config)#interface gigabitethernet0/0`	Moves to interface configuration mode.
`Edmonton(config-if)#ip flow ingress`	Enables NetFlow on the interface. Captures traffic that is being received by the interface.
`Edmonton(config-if)#ip flow egress`	Enables NetFlow on the interface. Captures traffic that is being transmitted by the interface.
`Edmonton(config-if)#exit`	Returns to global configuration mode.
`Edmonton(config)#ip flow-export destination` *ip_address udp_port*	Defines the IP Address of the workstation to which you want to send the NetFlow information as well as the UDP port on which the workstation is listening for the information.
`Edmonton(config)#ip flow-export version` *x*	Specifies the version format that the export packets used.

NOTE: NetFlow exports data in UDP in one of five formats: 1, 5, 7, 8, 9. Version 9 is the most versatile, but is not backward compatible with versions 5 or 8.

Verifying NetFlow

`Edmonton#show ip interface gigabitethernet0/0`	Displays information about the interface, including NetFlow as being either ingress or egress enabled
`Edmonton#show ip flow export`	Verifies status and statistics for NetFlow accounting data export
`Edmonton#show ip cache flow`	Displays a summary of NetFlow statistics on a Cisco IOS router

NOTE: The **show ip cache flow** command is useful for seeing which protocols use the highest volume of traffic and between which hosts this traffic flows.

Disabling Unneeded Services

Services that are not being used on a router can represent a potential security risk. If you do not need a specific service, you should disable it.

TIP: If a service is off by default, disabling it does not appear in the running configuration.

TIP: Do not assume that a service is disabled by default; you should explicitly disable all unneeded services, even if you think they are already disabled.

TIP: Depending on the IOS Software release, some services are on by default; some are on. Be sure to check the IOS configuration guide for your specific software release to determine the default state of the service.

Table 7-5 lists the services that you should disable if they are not being used.

TABLE 7-5 Disabling Unneeded Services

Service	Commands Used to Disable Service
DNS name resolution	`Edmonton(config)#no ip domain-lookup`
CDP (globally)	`Edmonton(config)#no cdp run`
CDP (on a specific interface)	`Edmonton(config-if)#no cdp enable`
NTP	`Edmonton(config-if)#ntp disable`
BOOTP server	`Edmonton(config)#no ip bootp server`
DHCP	`Edmonton(config)#no ip dhcp-server`
Proxy ARP	`Edmonton(config-if)no ip proxy-arp`
IP source routing	`Edmonton(config)#no ip source-route`
IP redirects	`Edmonton(config-if)#no ip redirects`
HTTP service	`Edmonton(config)#no ip http server`

Configuring Network Time Protocol

Most networks today are being designed with high performance and reliability in mind. Delivery of content is, in many cases, guaranteed by service level agreements (SLAs). Having your network display an accurate time is vital to ensuring that you have the best information possible when reading logging messages or troubleshooting issues.

NTP Configuration

`Edmonton(config)#ntp server 209.165.200.254`	Configures the Edmonton router to synchronize its clock to a public NTP server at address 209.165.200.254.
	NOTE: This command makes the Edmonton router an NTP client to the external NTP server.
	NOTE: A Cisco IOS router can be both a client to an external NTP server and an NTP server to client devices inside its own internal network.
	NOTE: When NTP is enabled on a Cisco IOS router is it enabled on all interfaces.
`Edmonton(config)#ntp server 209.165.200.234 prefer`	Specifies a preferred NTP server if multiple ones are configured.
	TIP: It is recommended to configure more than one NTP server.
`Edmonton(config)#ntp server 2001:DB8:0:0:8:800:200c:417A version 4`	Configures the Edmonton router to synchronize its clock to a public NTP server at address 2001:DB8:0:0:8:800:200c:417A.
	NOTE: Version 4 of NTP is also selected as it is the only NTP version with support for IPv6.
`Edmonton(config-if)#ntp disable`	Disables the NTP server function on a specific interface. The interface will still act as an NTP client.
	TIP: Use this command on interfaces connected to external networks.
`Edmonton(config)#ntp master` *`stratum`*	Configures the router to be an NTP master clock to which peers synchronize when no external NTP source is available. The *stratum* is an optional number between 1 and 15. When enabled, the default stratum is 8.
	NOTE: A reference clock (for example, an atomic clock) is said to be a stratum-0 device. A stratum-1 server is directly connected to a stratum-0 device. A stratum-2 server is connected across a network path to a stratum-1 server. The larger the stratum number (moving toward 15), the less authoritative that server is and the less accuracy it will have.
`Edmonton(config)#ntp max-associations 200`	Configures the maximum number of NTP peer-and-client associations that the router will serve. The range is 0 to 4,294,967,295. The default is 100.
`Edmonton(config)#access list 101 permit udp any host` *`a.b.c.d`* `eq ntp`	Creates an access list statement that will allow NTP communication for the NTP server at address *a.b.c.d*. This ACL should be placed in an inbound direction.

NOTE: When a local device is configured with the **ntp master** command, it can be identified by a syntactically correct but invalid IP address. This address will be in the form of 127.127.x.x. The master will synchronize with itself and uses the 127.127.x.x address to identify itself. This address will be displayed with the **show ntp associations** command and must be permitted via an access list if you are authenticating your NTP servers.

NTP Design

You have two different options in NTP design: flat and hierarchical. In a flat design, all routers will be peers to each other. Each router will be a both a client and a server with every other router. In a hierarchical model, there is a preferred order of routers that are servers and others that act as clients. You use the **ntp peer** command to determine the hierarchy.

TIP: Do not use the flat model in a large network, because with many NTP servers it can take a long time to synchronize the time.

`Edmonton(config)#ntp peer 172.16.21.1`	Configures an IOS device to synchronize its software clock to a peer at 172.16.21.1.
`Edmonton(config)#ntp peer 172.16.21.1 version 2`	Configures an IOS device to synchronize its software clock to a peer at 172.16.21.1 using version 2 of NTP. There are 3 versions of NTP (versions 2–4).

NOTE: Although Cisco IOS recognizes three version of NTP, versions 3 and 4 are most commonly used. Version 4 introduces support for IPv6 and is backward compatible with Version 3. NTPv4 also adds DNS support for IPv6.

NOTE: NTPv4 has increased security support using public key cryptography and X509 certificates.

NOTE: NTPv3 uses broadcast messages. NTPv4 uses multicast messages.

`Edmonton(config)#ntp peer 172.16.21.1 source loopback 0`	Configures an IOS device to synchronize its software clock to a peer at 172.16.21.1. The source IP address is the address of interface Loopback 0.
	TIP: Choose a loopback interface as your source for NTP because they never go down. ACL statements will also be easier to write as you will only require one line to allow or deny traffic.
`Edmonton(config)#ntp peer 172.16.21.1 source loopback 0 prefer`	Makes this peer the preferred peer that provides synchronization
`Edmonton(config)#ntp peer 2001:DB8:0:0:8:800:200c:417A version 4`	Configures the software clock to synchronize a peer or to be synchronized by a peer at address 2001:DB8:0:0:8:800:200c:417A.

Securing NTP

You can secure NTP operation using authentication and access lists.

Enabling NTP Authentication

`NTPServer(config)#ntp authentication-key 1 md5 NTPpa55word`	Defines an NTP authentication key. **1** = number of authentication key. Can be a number between 1 and 4,294,967,295. **md5** = using MD5 hash. This is the only option available on Cisco device. **NTPpa55word** = password associated with this key
`NTPServer(config)#ntp authenticate`	Enables NTP authentication.
`NTPServer(config)#ntp trusted-key 1`	Defines which keys are valid for NTP authentication. The key number here must match the key number you defined in the ntp **authentication-key** command.
`NTPClient(config)#ntp authentication-key 1 md5 NTPpa55word`	Defines an NTP authentication key.
`NTPClient(config)#ntp authenticate`	Enables NTP authentication.
`NTPClient(config)#ntp trusted-key 1`	Defines which keys are valid for NTP authentication. The key number here must match the key number you defined in the ntp **authentication-key** command.
`NTPClient(config)#ntp server 192.168.200.1 key 1`	Defines the NTP server that requires authentication at address 192.168.200.1 and identifies the peer key number as key 1.

NOTE: NTP does not authenticate clients; it only authenticates the source. That means that a device will respond to unauthenticated requests. Therefore, access lists should be used to limit NTP access.

NOTE: Once a device is synchronized to an NTP source, it will become an NTP server to any device that requests synchronization.

Limiting NTP Access with Access Lists

`Edmonton(config)#access-list 1 permit 10.1.0.0 0.0.255.255`	Defines an access list that permits only packets with a source address of 10.1.x.x.
`Edmonton(config)#ntp access-group peer 1`	Creates an access group to control NTP access and applies access list 1. The **peer** keyword enables the device to receive time requests and NTP control queries and to synchronize itself to servers specified in the access list.

`Edmonton(config)#ntp access-group serve 1`	Creates an access group to control NTP access and applies access list 1. The **serve** keyword enables the device to receive time requests and NTP control queries from the servers specified in the access list but not to synchronize itself to the specified servers.
`Edmonton(config)#ntp access-group serve-only 1`	Creates an access group to control NTP access and applies access list 1. The **serve-only** keyword enables the device to receive only time requests from servers specified in the access list.
`Edmonton(config)#ntp access-group query-only 1`	Creates an access group to control NTP access and applies access list 1. The **query-only** keyword enables the device to receive only NTP control queries from the servers specified in the access list.

NOTE: NTP access group options are scanned from least restrictive to most restrictive in the following order: **peer**, **serve**, **serve-only**, **query-only**. However, if NTP matches a deny ACL rule in a configured peer, ACL processing stops and does not continue to the next access group option.

Verifying NTP

`Edmonton#show ntp associations`	Displays the status of NTP associations.
`Edmonton#show ntp associations detail`	Displays detailed information about each NTP association.
`Edmonton#show ntp status`	Displays the status of the NTP. This command will show whether the router's clock has synchronized with the external NTP server.
`Edmonton#debug ip packets`	Checks to see whether NTP packets are received and sent.
`Edmonton#debug ip packet 1`	Limits debug output to ACL 1.
`Edmonton#debug ntp adjust`	Displays debug output for NTP clock adjustments.
`Edmonton#debug ntp all`	Displays all NTP debugging output.
`Edmonton#debug ntp events`	Displays all NTP debugging events.
`Edmonton#debug ntp packet`	Displays NTP packet debugging; lets you see the time that the peer/server gives you in a received packet.
`Edmonton#debug ntp packet detail`	Displays detailed NTP packet dump.

`Edmonton#debug ntp packet peer A.B.C.D` or `Edmonton#debug ntp packet peer X:X:X:X::X`	Displays debugging from NTP Peer at address A.B.C.D. Or Displays debugging from NTP peer at address X:X:X:X::X.

SNTP

NOTE: Simple NTP (SNTP) is a simplified, client-only version of NTP. SNTP can only receive the time from NTP servers; it cannot be used to provide time services to other systems.

`Router(config)#sntp server 209.165.200.187`	Configures a router to use SNTP to request and accept NTP traffic from a time server

TIP: Most SNTP commands, including authentication commands, are identical to NTP commands, with the only difference being the use of the **sntp** keyword rather than **ntp**.

NOTE: SNTP and NTP cannot coexist on the same machine because they use the same port number (UDP 123):

```
Edmonton(config)#sntp server 209.165.200.187

 %SNTP : Cannot configure SNTP as NTP is already running.
%SNTP : Unable to start SNTP process

Edmonton(config)#
```

Setting the Clock on a Router

NOTE: It is important to have your routers display the correct time for use with time stamps and other logging features.

If the system is synchronized by a valid outside timing mechanism, such as an NTP, or if you have a router with a hardware clock, you do not need to set the software clock. Use the software clock if no other time sources are available.

`Edmonton#calendar set 16:30:00 23 June 2014`	Manually sets the system hardware clock. Time is set using military (24-hour) format. The hardware clock runs continuously, even if the router is powered off or rebooted.
`Edmonton#show calendar`	Displays the hardware calendar.

`Edmonton(config)#clock calendar-valid`	Configures the system as an authoritative time source for a network based on its hardware clock.
	NOTE: Because the hardware clock is not as accurate as other time sources (it runs off of a battery), you should use this only when a more accurate time source (such as NTP) is not available.
`Edmonton#clock read-calendar`	Manually reads the hardware clock settings into the software clock.
`Edmonton#clock set 16:30:00 23 June 2014`	Manually sets the system software clock. Time is set using military (24-hour) format.
`Edmonton(config)#clock summer-time` *zone* `recurring` [*week day month hh:mm week day month hh:mm* [*offset*]] `Edmonton(config)#clock summer-time` *zone* `date` *date month year hh:mm date month year hh:mm* [*offset*] `Edmonton(config)#clock summer-time` *zone* `date` *month date year hh:mm month date year hh:mm* [*offset*]	Configures the system to automatically switch to summer time (daylight saving time). **NOTE:** Summer time is disabled by default. Arguments for the command are as follows: *zone*: Name of the time zone. **recurring**: Summer time should start and end on the corresponding specified days every year. **date**: Indicates that summer time should start on the first specific date listed in the command and end on the second specific date in the command. *week*: (Optional) Week of the month (1 to 5 or **last**). *day*: (Optional) Day of the week (Sunday, Monday, and so on). *date*: Date of the month (1 to 31). *month*: (Optional) Month (January, February, and so on). *year*: Year (1993 to 2035). *hh:mm*: (Optional) Time (military format) in hours and minutes. *offset*: (Optional) Number of minutes to add during summer time (default is 60).
`Edmonton(config)#clock timezone` *zone hours-offset* [*minutes-offset*]	Configures the time zone for display purposes. To set the time to coordinated universal time (UTC), use the **no** form of this command.
`Edmonton(config)#clock timezone PST -8`	Configures the time zone to pacific standard time, which is 8 hours behind UTC.

`Edmonton(config)#clock timezone NL -3 30`	Configures the time zone to Newfoundland time for Newfoundland, Canada, which is 3.5 hours behind UTC. *zone*: Name of the time zone to be displayed when standard time is in effect. See Tables 7-6 and 7-7 for common time zone acronyms. *hours-offset*: Hours difference from UTC. *minutes-offset*: (Optional) Minutes difference from UTC.
`Edmonton#clock update-calendar`	Updates the hardware clock from the software clock.
`Edmonton#show clock`	Displays the time and date from the system software clock.
`Edmonton#show clock detail`	Displays the clock source (NTP, hardware) and the current summer-time setting (if any).

Table 7-6 shows the common acronyms used for setting the time zone on a router.

TABLE 7-6 Common Time Zone Acronyms

Region/Acronym	Time Zone Name and UTC Offset
Europe	
GMT	Greenwich mean time, as UTC
BST	British summer time, as UTC +1 hour
IST	Irish summer time, as UTC +1 hour
WET	Western Europe time, as UTC
WEST	Western Europe summer time, as UTC +1 hour
CET	Central Europe time, as UTC +1
CEST	Central Europe summer time, as UTC +2
EET	Eastern Europe time, as UTC +2
EEST	Eastern Europe summer time, as UTC +3
MSK	Moscow time, as UTC +3
MSD	Moscow summer time, as UTC +4
United States and Canada	
AST	Atlantic standard time, as UTC –4 hours
ADT	Atlantic daylight time, as UTC –3 hours
ET	Eastern time, either as EST or EDT, depending on place and time of year
EST	Eastern standard time, as UTC –5 hours
EDT	Eastern daylight saving time, as UTC –4 hours

Region/Acronym	Time Zone Name and UTC Offset
CT	Central time, either as CST or CDT, depending on place and time of year
CST	Central standard time, as UTC –6 hours
CDT	Central daylight saving time, as UTC –5 hours
MT	Mountain time, either as MST or MDT, depending on place and time of year
MST	Mountain standard time, as UTC –7 hours
MDT	Mountain daylight saving time, as UTC –6 hours
PT	Pacific time, either as PST or PDT, depending on place and time of year
PST	Pacific standard time, as UTC –8 hours
PDT	Pacific daylight saving time, as UTC –7 hours
AKST	Alaska standard time, as UTC –9 hours
AKDT	Alaska standard daylight saving time, as UTC –8 hours
HST	Hawaiian standard time, as UTC –10 hours
Australia	
WST	Western standard time, as UTC +8 hours
CST	Central standard time, as UTC +9.5 hours
EST	Eastern standard/summer time, as UTC +10 hours (+11 hours during summer time)

Table 7-7 lists an alternative method for referring to time zones, in which single letters are used to refer to the time zone difference from UTC. Using this method, the letter *Z* is used to indicate the zero meridian, equivalent to UTC, and the letter *J* (Juliet) is used to refer to the local time zone. Using this method, the international date line is between time zones M and Y.

TABLE 7-7 Single-Letter Time Zone Designators

Letter Designator	Word Designator	Difference from UTC
Y	Yankee	UTC –12 hours
X	X-ray	UTC –11 hours
W	Whiskey	UTC –10 hours
V	Victor	UTC –9 hours
U	Uniform	UTC –8 hours
T	Tango	UTC –7 hours
S	Sierra	UTC –6 hours
R	Romeo	UTC –5 hours
Q	Quebec	UTC –4 hours
P	Papa	UTC –3 hours

Letter Designator	Word Designator	Difference from UTC
O	Oscar	UTC –2 hours
N	November	UTC –1 hour
Z	Zulu	Same as UTC
A	Alpha	UTC +1 hour
B	Bravo	UTC +2 hours
C	Charlie	UTC +3 hours
D	Delta	UTC +4 hours
E	Echo	UTC +5 hours
F	Foxtrot	UTC +6 hours
G	Golf	UTC +7 hours
H	Hotel	UTC +8 hours
I	India	UTC +9 hours
K	Kilo	UTC +10 hours
L	Lima	UTC +11 hours
M	Mike	UTC +12 hours

Using Time Stamps

`Edmonton(config)#service timestamps`	Adds a time stamp to all system logging messages
`Edmonton(config)#service timestamps debug`	Adds a time stamp to all debugging messages
`Edmonton(config)#service timestamps debug uptime`	Adds a time stamp along with the total uptime of the router to all debugging messages
`Edmonton(config)#service timestamps debug datetime localtime`	Adds a time stamp displaying the local time and the date to all debugging messages
`Edmonton(config)#no service timestamps`	Disables all time stamps

Configuration Example: NTP

Figure 7-1 shows the network topology for the configuration that follows, which demonstrates how to configure NTP using the commands covered in this chapter.

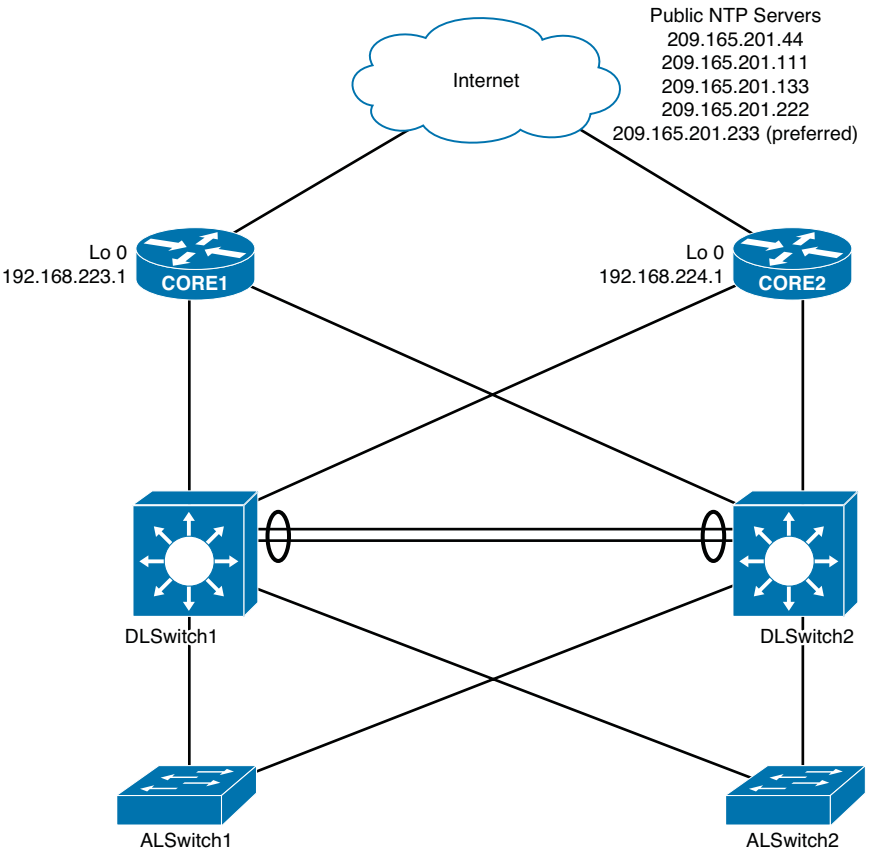


Figure 7-1 Network Topology for NTP Configuration

Core1 Router

`Core1(config)#ntp server 209.165.201.44`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.44.
`Core1(config)#ntp server 209.165.201.111`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.111.
`Core1(config)#ntp server 209.165.201.133`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.133.
`Core1(config)#ntp server 209.165.201.222`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.222.
`Core1(config)#ntp server 209.165.201.233 prefer`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.233. This is the preferred NTP server.
`Core1(config)#ntp max-associations 200`	Configures the maximum number of NTP peer-and-client associations that the router will serve.

`Core1(config)#clock timezone EDT -5`	Sets time zone to eastern daylight time.
`Core1(config)#clock summer-time EDT recurring 2 Sun Mar 2:00 1 Sun Nov 2:00`	Configures the system to automatically switch to summer time and to repeat on the same day.
`Core1(config)#ntp master 10`	Configures the router to server as a master clock if the external NPT server is not available.
`Core1(config)#access-list 1 permit 127.127.1.1`	Sets access list to permit packets coming from 127.127.1.1.
`Core1(config)#access-list 2 permit 192.168.0.0 0.0.255.255`	Sets access list to permit packets coming from 192.168.x.x.
`Core1(config)#ntp access-group peer 1`	Configures Core1 to peer with any devices identified in access list 1.
`Core1(config)#ntp access-group serve-only 2`	Configures Core1 to receive only time requests from devices specified in the access list.

Core2 Router

`Core2(config)#ntp server 209.165.201.44`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.44.
`Core2(config)#ntp server 209.165.201.111`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.111.
`Core2(config)#ntp server 209.165.201.133`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.133.
`Core2(config)#ntp server 209.165.201.222`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.222.
`Core2(config)#ntp server 209.165.201.233 prefer`	Configures router to synchronize its clock to a public NTP server at address 209.165.201.233. This is the preferred NTP server.
`Core2(config)#ntp max-associations 200`	Configures the maximum number of NTP peer-and-client associations that the router will serve.
`Core2(config)#clock timezone EDT -5`	Sets time zone to eastern daylight time.
`Core2(config)#clock summer-time EDT recurring 2 Sun Mar 2:00 1 Sun Nov 2:00`	Configures the system to automatically switch to summer time and to repeat on the same day.
`Core2(config)#ntp master 10`	Configures the router to server as a master clock if the external NPT server is not available.

`Core2(config)#access-list 1 permit 127.127.1.1`	Sets access list to permit packets coming from 127.127.1.1.
`Core2(config)#access-list 2 permit 192.168.0.0 0.0.255.255`	Sets access list to permit packets coming from 192.168.x.x.
`Core2(config)#ntp access-group peer 1`	Configures Core2 to peer with any devices identified in access list 1.
`Core2(config)#ntp access-group serve-only 2`	Configures Core2 to receive only time requests from devices specified in the access list.

DLSwitch1

`DLSwitch1(config)#ntp server 192.168.223.1`	Configures DLSwitch1 to synchronize its clock to a NTP server at address 192.168.223.1
`DLSwitch1(config)#ntp server 192.168.224.1`	Configures DLSwitch1 to synchronize its clock to a NTP server at address 192.168.224.1
`DLSwitch1(config)#clock timezone EDT -5`	Sets time zone to eastern daylight time
`DLSwitch1(config)#clock summer-time EDT recurring 2 Sun Mar 2:00 1 Sun Nov 2:00`	Configures the system to automatically switch to summer time and to repeat on the same day

DLSwitch2

`DLSwitch2(config)#ntp server 192.168.223.1`	Configures DLSwitch2 to synchronize its clock to a NTP server at address 192.168.223.1
`DLSwitch2(config)#ntp server 192.168.224.1`	Configures DLSwitch2 to synchronize its clock to a NTP server at address 192.168.224.1
`DLSwitch2(config)#clock timezone EDT -5`	Sets time zone to eastern daylight time
`DLSwitch2(config)#clock summer-time EDT recurring 2 Sun Mar 2:00 1 Sun Nov 2:00`	Configures the system to automatically switch to summer time and to repeat on the same day

ALSwitch1

`ALSwitch1(config)#ntp server 192.168.223.1`	Configures ALSwitch1 to synchronize its clock to a NTP server at address 192.168.223.1
`ALSwitch1(config)#ntp server 192.168.224.1`	Configures ALSwitch1 to synchronize its clock to a NTP server at address 192.168.224.1

`ALSwitch1(config)#clock timezone EDT -5`	Sets time zone to eastern daylight time
`ALSwitch1(config)#clock summer-time EDT recurring 2 Sun Mar 2:00 1 Sun Nov 2:00`	Configures the system to automatically switch to summer time and to repeat on the same day

ALSwitch2

`ALSwitch2(config)#ntp server 192.168.223.1`	Configures ALSwitch2 to synchronize its clock to a NTP server at address 192.168.223.1
`ALSwitch2(config)#ntp server 192.168.224.1`	Configures ALSwitch2 to synchronize its clock to a NTP server at address 192.168.224.1
`ALSwitch2(config)#clock timezone EDT -5`	Sets time zone to eastern daylight time
`ALSwitch2(config)#clock summer-time EDT recurring 2 Sun Mar 2:00 1 Sun Nov 2:00`	Configures the system to automatically switch to summer time and to repeat on the same day

Authentication of Routing Protocols

Security breaches may occur in your network by having unwanted parties interfere with your routers exchanging routes and destination information. Having your routers authenticate with each other before exchanging information is a recommended security practice.

Authentication Options for Different Routing Protocols

Table 7-8 shows the different authentication options that are available with different routing protocols.

TABLE 7-8 Authentication Options for Different Routing Protocols

Routing Protocol	Plain-text Authentication	Hashing Authentication (MD5)	Hashing Authentication (SHA)	Key Chain Support
BGP	No	Yes	No	No
EIGRP	No	Yes	Yes	Yes
OSPFv2	Yes	Yes	Yes	Yes
OSPFv3	No	Yes	Yes	No
RIPv2	Yes	Yes	No	Yes

NOTE: EIGRP support for Secure Hash (SHA) was introduced in Cisco IOS 15 together with named EIGRP configuration mode.

NOTE: EIGRP SHA does not use key chains.

NOTE: OSPFv2 uses built-in authentication mechanisms. OSPFv3 relies on IPv6 native security capabilities and the native security stack, which included IPsec.

NOTE: OSPFv2 key chains are supported on Nexus but not in IOS.

NOTE: RIPng does not support authentication; it relies on IPsec within IPv6.

Authentication for EIGRP

Authentication for routers using EIGRP relies on the use of predefined passwords.

NOTE: EIGRP IPv4 and EIGRP IPv6 use the same commands for authentication.

Configuring EIGRP Authentication

`Router(config)#key chain romeo`	Identifies a key chain. The name must match the name configured in interface configuration mode.
`Router(config-keychain)#key 1`	Identifies the key number.
	NOTE: The range of keys is from 0 to 2,147,483,647. The key identification numbers do not need to be consecutive. There must be at least 1 key defined on a key chain.
`Router(config-keychain-key)#key-string shakespeare`	Identifies the key string.
	NOTE: The string can contain from 1 to 80 uppercase and lowercase alphanumeric characters, except that the first character cannot be a number.
`Router(config-keychain-key)#accept-lifetime` *start-time* {**infinite** | end-time | **duration** *seconds*}	Optionally specifies the period during which the key can be received.
	NOTE: The default start time and the earliest acceptable date is January 1, 1993. The default end time is an infinite time period.
`Router(config-keychain-key)#send-lifetime` *start-time* {**infinite** | end-time | **duration** *seconds*}	Optionally specifies the period during which the key can be sent.
	NOTE: The default start time and the earliest acceptable date is January 1, 1993. The default end time is an infinite period.

`Router(config)#interface gigabitethernet0/0`	Enters interface configuration mode.
`Router(config-if)#ip authentication mode eigrp 100 md5`	Enables message digest 5 (MD5) authentication in EIGRP packets over the interface.
`Router(config-if)#ip authentication key-chain eigrp 100 romeo`	Enables authentication of EIGRP packets. **romeo** is the name of the key chain.
`Router(config-if)#exit`	Returns to global configuration mode.

NOTE: For the start time and the end time to have relevance, ensure that the router knows the correct time. Recommended practice dictates that you run NTP or some other time-synchronization method if you intend to set lifetimes on keys.

Configuring Authentication in Named EIGRP

NOTE: EIGRP support for SHA was introduced in Cisco IOS 15 together with named EIGRP configuration mode.

NOTE: Both MD5 and SHA can be used in either of IPv4 or IPv6. Not all permutations are shown in the following example.

`Router(config)#router eigrp TEST`	Creates a named EIGRP virtual instance called TEST.
`Router(config-router)#address-family ipv4 autonomous-system 1`	Enables the IPv4 address family and starts EIGRP autonomous system 1.
`Router(config-router-af)#af-interface gigabitethernet 0/0`	Moves the router into the address family interface configuration mode for interface Gigabit Ethernet 0/0.
`Router(config-router-af-interface)#authentication key-chain romeo`	Identifies a key chain.
`Router(config-router-af-interface)#authentication mode md5`	Enables message digest 5 (MD5) authentication in EIGRP packets over the interface.
`Router(config-router-af-interface)#exit-af-interface`	Exits from address family interface configuration mode.
`Router(config-router-af)#exit-address-family`	Exits address family configuration mode.
`Router(config-router)#address-family ipv6 autonomous-system 1`	Enables the IPv6 address family and starts EIGRP autonomous system 1.
`Router(config-router-af)#af-interface gigabitethernet 0/0`	Moves the router into the address family interface configuration mode for interface Gigabit Ethernet 0/0.

`Router(config-router-af-interface)#authentication key-chain romeo`	Identifies a key chain.
`Router(config-router-af-interface)#authentication mode hmac-sha-256 0 password1`	Enables advanced SHA authentication in EIGRP packets over the interface. The password used is **password1**.
`Router(config-router-af-interface)#exit-af-interface`	Exits from address family interface configuration mode.
`Router(config-router-af)#exit-address-family`	Exits address family configuration mode.
`Router(config-router)#exit`	Exits router protocol configuration mode.
`Router(config)#key chain romeo`	Identifies a key chain. Name must match the name configured in interface configuration mode.
`Router(config-keychain)#key 1`	Identifies the key number.
`Router(config-keychain-key)#key-string shakespeare`	Identifies the key string.
`Router(config-keychain-key)#accept-lifetime start-time {infinite | end-time | duration seconds}`	Optionally specifies the period during which the key can be received.
`Router(config-keychain-key)#send-lifetime start-time {infinite | end-time | duration seconds}`	Optionally specifies the period during which the key can be sent.

Verifying and Troubleshooting EIGRP Authentication

`Router#show ip eigrp neighbor`	Displays EIGRP neighbor table. Incorrect authentication configuration will prevent neighbor relationships from forming.
`Router#show ipv6 eigrp neighbor`	Displays EIGRP IPv6 neighbor table. Incorrect authentication configuration will prevent neighbor relationships from forming.
`Router#show key chain`	Displays key chains created on the router.
`Router#debug eigrp packet`	Displays output about EIGRP packets. Incorrect key string configuration will cause failures, which will be shown in this output.

Authentication for OSPF

Authentication for routers using OPSF also relies on the use of predefined passwords.

Configuring OSPFv2 Authentication: Simple

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#area 0 authentication`	Enables simple authentication; password will be sent in clear text.
`Router(config-router)#exit`	Returns to global configuration mode.
`Router(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#ip ospf authentication`	Another way to enable authentication if it has not been set up in router configuration mode shown earlier.
`Router(config-if)#ip ospf authentication-key clear`	Sets key (password) to clear.
	NOTE: The password can be any continuous string of characters that can be entered from the keyboard, up to 8 characters in length. To be able to exchange OSPF information, all neighboring routers on the same network must have the same password.
	NOTE: In Cisco IOS Software release 12.4, the router will give a warning if you try to configure a password longer than 8 characters; only the first 8 characters will be used. Some earlier Cisco IOS releases did not provide this warning.

Configuring OSPFv2 Authentication: Using MD5 Encryption

`Router(config)#router ospf 1`	Starts OSPF process 1.
`Router(config-router)#area 0 authentication message-digest`	Enables authentication with MD5 password encryption.
`Router(config-router)#exit`	Returns to global configuration mode.
`Router(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#ip ospf authentication message-digest`	Another way to enable authentication if it has not been set up in router configuration mode shown earlier.
`Router(config-if)#ip ospf message-digest-key 1 md5 secret`	**1** is the key ID. This value must be the same as that of your neighboring router. **md5** indicates that the MD5 hash algorithm will be used. **secret** is the key (password) and must be the same as that of your neighboring router.
	NOTE: If the **service password-encryption** command is not used when implementing OSPF MD5 authentication, the MD5 secret will be stored as plain text in NVRAM.

	NOTE: In Cisco IOS Software Release 12.4, the router will give a warning if you try to configure a password longer than 16 characters; only the first 16 characters will be used. Some earlier Cisco IOS releases did not provide this warning.

TIP: It is recommended that you keep no more than one key per interface. Every time you add a new key, you should remove the old key to prevent the local system from continuing to communicate with a hostile system that knows the old key.

NOTE: If the **service password-encryption** command is not used when configuring OSPF authentication, the key will be stored as plain text in the router configuration. If you use the **service password-encryption** command, there will be an encryption type of 7 specified before the encrypted key.

Configuring OSPFv2 Authentication: Using SHA Encryption

`Router(config)#key chain samplechain`	Specifies the key chain name and enters into key chain configuration mode.
`Router(config-keychain)#key 1`	Specifies the key identifier and enters key chain key configuration mode. The range is from 1 to 255.
`Router(config-keychain-key)#key-string ThisIsASampleKey54321`	Specifies the key string
`Router(config-keychain-key)#cryptographic-algorithm hmac-sha-256`	Configures the key with the specified cryptographic algorithm.
`Router(config-keychain-key)#send-lifetime local 10:00:00 15 August 2014 infinite`	Sets the time period during which an authentication key on a key chain is valid to be sent during key exchange with another device.
`Router(config-keychain-key)#exit`	Exits key-chain key configuration mode and returns to key chain configuration mode.
`Router(config-keychain)#exit`	Exits key chain configuration mode and returns to global configuration mode.
`Router(config)#interface gigabitethernet0/0`	Enters into interface configuration mode.
`Router(config-if)#ip ospf authentication key-chain samplechain`	Specifies the key chain for the interface.

Configuring OSPFv3 Authentication and Encryption

TIP: OSPFv3 requires the use of IPsec to enable authentication. Crypto images are therefore needed for authentication, as they are the only images that include the IPsec application programming interface (API) needed for use with OSPFv3.

NOTE: Authentication and encryption does not need to be done on both the interface and on the area, but rather only in one location. The following section shows both methods.

`Router(config)#interface gigabitethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#ipv6 ospf authentication ipsec spi 500 md5 0 1234567890abcdef1234567890abcdef`	Applies authentication policy to the interface. **spi** (security policy index) is analogous to key numbers in a key chain but is communicated via the Authentication Header (AH). The SPI is a number between 256 and 4,294,967,295. **md5** = using the MD5 hash algorithm. SHA1 is also an option.
	NOTE: The key string length is precise; it must be 32 hex digits for MD5 or 40 for SHA1.
`Router(config-if)#ospfv3 authentication ipsec spi 500 md5 0 1234567890abcdef1234567890abcdef`	Alternative way of applying authentication policy to the interface.
`Router(config-if)#ipv6 ospf encryption ipsec spi 1001 esp null sha1 123456789A123456789B123456789C123456789D`	Specifies the encryption type for the interface.
`Router(config-if)#ospfv3 encryption ipsec spi 1001 esp null md5 0 1234567890abcdef1234567890abcdef`	Alternative way of specifying the encryption type for the interface.
`Router(config-if)#exit`	Returns to global configuration mode.
`Router(config)#router ospfv3 1`	Moves to routing protocol configuration mode.
`Router(config-router)#area 0 authentication ipsec spi sha1 12345678901234567890123456789012345678901234567890`	Applies authentication policy to an entire area.
`Router(config-router)#area 0 encryption ipsec spi 500 esp null md5 1aaa2bbb3ccc4ddd5eee6fff7aaa8bbb`	Enables encryption for the entire area.
`Router(config-router)#exit`	Returns to global configuration mode.

Verifying OSPFv2 and OSPFv3 Authentication

`Router#show ip ospf neighbor`	Displays OSPF neighbor table. Incorrect authentication configuration will prevent neighbor relationships from forming.
`Router#show ip route ospf`	Displays the OSPF routes in the routing table. Incorrect authentication configuration will prevent routes from being inserted into the routing table.
`Router#show ospfv3 neighbor`	Displays the OSPFv3 neighbor table.
`Router#show ipv6 route ospf`	Displays the OSPFv3 routes in the routing table.
`Router#show ip ospf interface gigabitethernet0/0`	Verifies authentication setup on a specific interface.
`Router#show crypto ipsec sa interface gigabitethernet0/0`	Displays IPsec security associations on a specific interface.
`Router#debug ip ospf adj`	Displays information about OSPF adjacencies and authentication for IPv4.
`Router#debug ipv6 ospf adj`	Displays information about OSPF adjacencies and authentication for IPv6.

Authentication for BGP and BGP for IPv6

Authentication for routers using Border Gateway Protocol (BGP) also relies on the use of predefined passwords and uses MD5.

Configuring Authentication Between BGP Peers

`Router(config)#router bgp 65100`	Enters routing protocol configuration mode.
`Router(config-router)neighbor 209.165.202.130 remote-as 65000`	Defines a BGP peer at IP address 209.165.202.130.
`Router(config-router)#neighbor 209.165.202.130 password P@55word`	Enables MD5 authentication on a TCP connection with peer at IP address 209.165.202.130. The password is **P@55word**.
`Router(config-router)#neighbor 2001:db8:0:10::1 password P@55word`	Enables MD5 authentication on a TCP connection with peer at IPv6 address 2001:db8:0:10::1. The password is **P@55word**.
	NOTE: To avoid losing your peer relationship, the same password must be configured on your remote peer before the hold-down timer expires, which has a default setting of 180 seconds.

Verifying BGP and BGP for IPv6 Authentication

`Router#show ip bgp summary`	Displays summary of BGP neighbor status
`Router#show ip bgp neighbors`	Displays detailed information on TCP and BGP neighbor connections
`Router#show bgp ipv6 unicast summary`	Displays the status of all IPv6 BGP connections.
`Router#show bgp ipv6 unicast neighbors`	Displays information about IPv6 BGP connections to neighbors

CHAPTER 8

Basic Concepts and Network Design

This chapter provides information about the following topics:

- Hierarchical model (Cisco enterprise campus architecture)
- Verifying switch content-addressable memory
- Switching Database Manager templates
- Configuring SDM templates
  - Verifying SDM templates
- LLDP (802.1AB)
  - Configuring LLDP
  - Verifying LLDP
- Power over Ethernet
  - Configuring PoE
  - Verifying PoE

CAUTION: Your hardware platform or software release might not support all the commands documented in this chapter. Please refer to the Cisco website for specific platform and software release notes.

Hierarchical Model (Cisco Enterprise Campus Architecture)

Figure 8-1 illustrates the hierarchical model at a high level as applied to a campus network design.

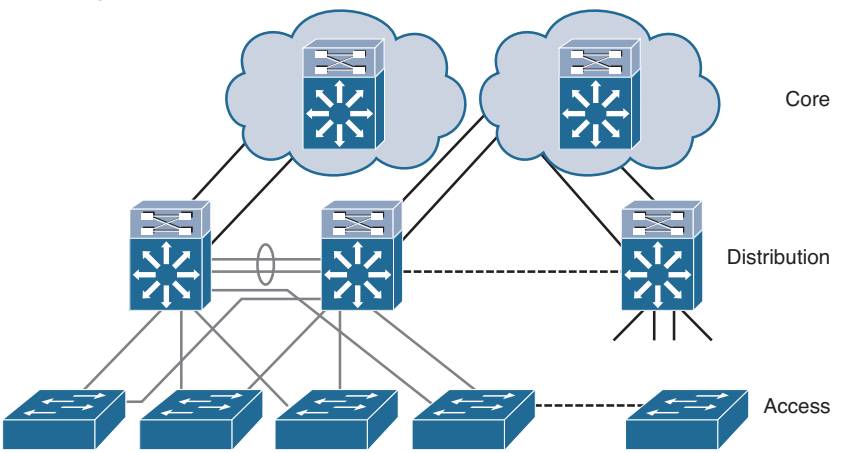


Figure 8-1 High-Level Example of the Hierarchical Model as Applied to a Campus Network

Verifying Switch Content-Addressable Memory

`Switch#show mac address-table`	Displays the content of the MAC address table (CAM) of the switch.
`Switch#show mac address-table interface fastethernet0/1`	Displays MAC addresses for a specific interface.
`Switch#show mac address-table address aabb.ccdd.eeff`	Display information for a specific MAC address.
`Switch#show mac address-table vlan 5`	Display MAC addresses for a specific VLAN.
`Switch#show mac address-table aging-time`	Display aging time for dynamic addresses for all VLANs.
`Switch(config)#mac address-table aging-time 400`	Sets the length of time (in seconds) that a dynamic entry remains in the MAC address table after the entry is used or updated. The aging time applies to all VLANs. The default value is 300 seconds. The range is 10 to 1,000,000 seconds.
	TIP: A value of 0 disables aging.

Switching Database Manager Templates

Cisco Switching Database Manager (SDM) templates are used to configure system resources in the switch to optimize support for specific features. The most common action is to change from the default template to the dual-stack template. IPv6 functionalities are not supported with the default template, only with the templates that specifically mention IPv6.

Configuring SDM Templates

`Switch(config)#sdm prefer access`	Provides maximum system usage for access control lists (ACLs). Use this template if you have a large number of ACLs.
`Switch(config)#sdm prefer default`	Gives balance to all functions. This is the default value.
`Switch(config)#sdm prefer dual-ipv4-and-ipv6 default`	Provides balance to IPv4 and IPv6 Layer 2 and Layer 3 functionality.
`Switch(config)#sdm prefer dual-ipv4-and-ipv6 routing`	Provides maximum system usage for IPv4 and IPv6 routing, including IPv4 policy-based routing.
`Switch(config)#sdm prefer dual-ipv4-and-ipv6 vlan`	Provides maximum system usage for IPv4 and IPv6 VLANs.

`Switch(config)#sdm prefer routing`	Provides maximum system usage for unicast routing. You would typically use this template for a router or aggregator in the middle of a network.
`Switch(config)#sdm prefer vlan`	Provides maximum system usage for VLANs. This template maximizes system resources for use as a Layer 2 switch with no routing.
`Switch(config)#sdm prefer indirect-ipv4-and-ipv6-routing`	Allows more entries for IPv4 and IPv6 summary or indirect routes, and fewer entries for IPv4 and IPv6 policy-based routing, quality of service (QoS), and ACLs.

NOTE: You must save the configuration and then reload the switch for the change to take effect. If you enter the **show sdm prefer** command before you enter the **reload** privileged EXEC command, the **show sdm prefer** command shows the template currently in use and the template that will become active after a reload.

Verifying SDM Templates

`Switch#show sdm prefer`	Displays information about the current SDM template with an approximate resource allocation per feature
`Switch#show platform tcam utilization`	Displays how much TCAM (ternary content-addressable memory) has now been utilized and how much is available (command only available on Catalyst 3750 platform)

TIP: Similar to the **sdm prefer** command, the **show sdm prefer** command can be used with the **default**, **access**, **routing**, **vlan**, and **dual-ipv4-and-ipv6** keywords to verify how each templates attributes resources.

NOTE: Not all **sdm prefer** options are supported on all switch platforms. Table 8-1 lists the options for the 2960, 3650, and 3750 platforms.

TABLE 8-1 SDM Options Available by Platform

Platform	Options Available
Catalyst 2960 and Catalyst 2960-C Fast Ethernet switch	**default** **dual-ipv4-and-ipv6** **lanbase-routing** **qos**
Catalyst 2960-S	**default** **lanbase-routing**

Platform	Options Available
Catalyst 2960-C Gigabit Ethernet switch	**default**
Catalyst 3650	**access** **default** **dual-ipv4-and-ipv6** **routing** **vlan**
Catalyst 3560-C	**default**
Catalyst 3560-X	**access** **default** **dual-ipv4-and-ipv6** **indirect-ipv4-and-ipv6-routing** **routing** **vlan**
Catalyst 3750-X	**access** **default** **dual-ipv4-and-ipv6** **indirect-ipv4-and-ipv6-routing** **routing** **vlan**
Catalyst 3750-E	**access** **default** **dual-ipv4-and-ipv6** **routing** **vlan**

LLDP (802.1AB)

Link Layer Discovery Protocol (LLDP) is an industry standard alternative to Cisco Discovery Protocol (CDP).

Configuring LLDP

`Switch(config)#lldp run`	Enables LLDP globally on the switch.
`Switch(config)#no lldp run`	Disables LLDP globally on the switch.

`Switch(config)#lldp holdtime 180`	Specifies the amount of time a receiving device should hold the information sent by another device before discarding it. The default value is 120 seconds. The range is 0 to 65,535 seconds.
`Switch(config)#lldp timer 60`	Sets the transmission frequency of LLDP updates in seconds. The default value is 30 seconds. The range is 5 to 65,534 seconds.
`Switch(config)#interface fastethernet0/1`	Specifies the interface on which you are enabling or disabling LLDP and enters interface configuration mode.
`Switch(config-if)#lldp transmit`	Enables the interface to send LLDP.
`Switch(config-if)#lldp receive`	Enables the interface to receive LLDP.
`Switch(config-if)#no lldp transmit`	No LLDP packets are sent on the interface.
`Switch(config-if)#no lldp receive`	No LLDP packets are received on the interface.

Verifying LLDP

`Switch#clear lldp counters`	Reset the traffic counters to 0.
`Switch#clear lldp table`	Delete the LLDP table of information about neighbors.
`Switch#show lldp`	Display global information, such as frequency of transmissions, the holdtime for packets being sent, and the delay time for LLDP to initialize on an interface.
`Switch#show ldp entry` *entry-name*	Display information about a specific neighbor. You can enter an asterisk (*) to display all neighbors, or you can enter the name of the neighbor about which you want information.
`Switch#show lldp interface` [*interface-id*]	Display information about interfaces where LLDP is enabled. You can limit the display to the interface about which you want information.
`Switch#show lldp neighbors` [*interface-id*] `[detail]`	Display information about neighbors, including device type, interface type and number, holdtime settings, capabilities, and port ID.
`Switch#show lldp traffic`	Display LLDP counters, including the number of packets sent and received, number of packets discarded, and number of unrecognized TLV (Type Length Value) fields.

Power over Ethernet

You can turn on Power over Ethernet (PoE) support at the port level. A device not needing any PoE can still be connected to that port; power is supplied only if the device requires it. The amount of power that is supplied will be automatically detected.

Configuring PoE

`Switch(config)#interface fastethernet0/1`	Enters interface configuration mode.
`Switch(config-if)#power inline auto`	Enables powered-device detection. If enough power is available, automatically allocates power to the PoE port after device detection.
`Switch(config-if)#power inline never`	Disables device detection, and disables power to the port.
`Switch(config-if)#power inline static max` *max-wattage*	Enables powered-device detection. Pre-allocates (reserves) power for a port before the switch discovers the powered device. The range is 4000 to 15,400 milliwatts on a Catalyst 2960 switch, and 4000 to 30,000 milliwatts on a Catalyst 2960-S switch. If no value is specified, the maximum is allowed. See the notes that follow here for other ranges.

NOTE: The default power output on a Catalyst 2960 is 15.4 W and 30 W on a Catalyst 2960-S switch.

NOTE: The default power output on a Catalyst 3560 PoE switch is 15.4 W and 30 W on a Catalyst 3560 PoE+ switch.

NOTE: The default power output on a Catalyst 3750-X and 3560-X PoE switch 30 W.

Verifying PoE

`Switch#show power inline`	Displays the PoE status for all PoE ports in the switch
`Switch#show power inline fastethernet0/1`	Displays the PoE status for a specific port in the switch
`Switch#show power inline consumption`	Displays the power allocated to devices connected to PoE ports
`Switch#show controllers power inline`	Displays the values in the registers of the specified PoE controller
`Switch(config)#interface fastethernet0/1`	Enters interface configuration mode
`Switch(config-if)#logging event power-inline-status`	Enables the logging of PoE events for a specific interface

CHAPTER 9

Campus Network Architecture

This chapter provides information about the following topics:

- Virtual LANs
  - Creating static VLANs
  - Normal-range static VLAN configuration
  - Extended-range static VLAN-configuration
  - Assigning ports to data and voice VLANs
  - Using the range command
  - Dynamic Trunking Protocol (DTP)
  - Setting the trunk encapsulation type and allowed VLANs
  - Verifying VLAN information
  - Saving VLAN configurations
  - Erasing VLAN configurations
  - Verifying VLAN trunking
  - VLAN Trunking Protocol
  - Using global configuration mode
  - Verifying VTP
  - Configuration example: VLANs
- Layer 2 Link Aggregation
  - Link Aggregation Interface Modes
  - Guidelines for Configuring Link Aggregation
  - Guidelines for configuring EtherChannel
  - Configuring L2 EtherChannel
  - Configuring L3 EtherChannel
  - Verifying EtherChannel
  - Configuring EtherChannel load balancing
  - Configuration example: PAgP EtherChannel
- Implementing DHCP for IPv4
  - Configuring basic DHCP server for IPv4
  - Configuring DHCP manual IP assignment for IPv4

- Configuring DHCP Relay IPv4
- Verifying DHCP for IPv4

- Implementing DHCP for IPv6
- Configuring DHCPv6 server
- Configuring DHCPv6 client
- Configuring DHCPv6 relay agent
- Verifying DHCPv6

CAUTION: Your hardware platform or software release might not support all the commands documented in this chapter. Please refer to the Cisco website for specific platform and software release notes.

Virtual LANs

A VLAN is a switched network that logically segments by function, project teams, or applications, without regard to the physical locations of the users. VLANs are the Layer 2 (L2) partitioning of a physical switch into two or more virtual switches. Ports assigned to one VLAN are in single broadcast domain and are L2 forwarded only within that broadcast domain. Each VLAN is considered its own logical network where any traffic destined for outside the logical network must be forwarded by a router. Each VLAN can support its own instance of spanning tree. VLANs can be extended across multiple interconnected switches by tagging the VLAN number on each Ethernet frame transmitted or received between them, IEEE 802.1Q.

Creating Static VLANs

Static VLANs occur when a switch port is manually assigned by the network administrator to belong to a VLAN. Each port is associated with a specific VLAN. By default, all ports are originally assigned to VLAN 1. VLANs are created using VLAN configuration mode.

NOTE: VLAN database mode has been deprecated in IOS Version 15.

Normal-Range static VLAN Configuration

`Switch(config)#vlan 3`	Creates VLAN 3 and enters VLAN configuration mode for further definitions.
`Switch(config-vlan)#name Engineering`	Assigns a name to the VLAN. The length of the name can be from 1 to 32 characters. The default name of a VLAN is VLAN*xxxx*, where *xxxx* is the VLAN number.

`Switch(config-vlan)#exit`	Applies changes, increases the revision number by 1, and returns to global configuration mode.
`Switch(config)#`	

Extended-Range static VLAN Configuration

`Switch#configure terminal`	Enters global configuration mode.
`Switch(config)#vtp mode transparent`	Configures the switch for VTP transparent mode, disabling VTP.
	NOTE: This step is not required for VTP Version 3.
`Switch(config)#vlan 2000`	Creates VLAN 2000 and enters VLAN configuration mode for further definitions.
`Switch(config-vlan)#exit`	Applies changes, increases the revision number by 1, and returns to global configuration mode.
`Switch(config)#`	

NOTE: This method is the only way to configure extended-range VLANs (VLAN IDs from 1006 to 4094).

NOTE: The VTP revision number is increased by one each time a VLAN is created or changed, except when the switch is in transparent mode.

Assigning Ports to Data and Voice VLANs

`Switch(config)#interface fastethernet 0/1`	Moves to interface configuration mode
`Switch(config-if)#switchport mode access`	Sets the port to access mode
`Switch(config-if)#switchport access vlan 10`	Assigns this port to data VLAN 10
`Switch(config-if)#switchport voice vlan 11`	Assigns this port to include tagged voice frames in VLAN 11

NOTE: When the **switchport mode access** command is used, the port will operate as a nontrunking single VLAN interface that transmits and receives nonencapsulated frames. An access port can belong to only one VLAN.

NOTE: When the **switchport voice** command is used together with the **switchport access** command, a mini-trunk is created allowing two VLANs on the port, one for voice traffic and one for all other traffic. The voice traffic is forwarded in 802.1Q tagged frames and the remaining nonvoice VLAN has no 802.1Q tagging (native VLAN). The internal mini-switch in a Cisco VoIP phone will pass untagged frames to an attached PC and forward 802.1Q tagged VoIP traffic with a differentiated services code point (DSCP) quality of service (QoS) value of EF (or Expedited Forwarding) to the switch port. In the case of a mini-trunk, the switch port can belong to two VLANs.

Using the range Command

The **interface range** command is one of the many useful commands that is not part of the SWITCH exam.

`Switch(config)#interface range fastethernet 0/1 -9`	Enables you to set the same configuration parameters on multiple ports at the same time.
	NOTE: There is a space before and after the hyphen in the **interface range** command.
`Switch(config-if-range)#switchport mode access`	Sets ports 1–9 as access ports.
`Switch(config-if-range)#switchport access vlan 10`	Assigns ports 1–9 to native data VLAN 10.
`Switch(config-if-range)#switchport voice vlan 11`	Assigns ports 1–9 to include tagged voice frames in VLAN 11.

Dynamic Trunking Protocol

`Switch(config)#interface fastethernet 0/1`	Moves to interface configuration mode
`Switch(config-if)#switchport mode dynamic desirable`	Makes the interface actively attempt to convert the link to a trunk link.
	NOTE: With the **switchport mode dynamic desirable** command set, the interface will become a trunk link if the neighboring interface is set to **trunk**, **desirable**, or **auto**.
`Switch(config-if)#switchport mode dynamic auto`	Makes the interface able to convert into a trunk link.
	NOTE: With the **switchport mode dynamic auto** command set, the interface will become a trunk link if the neighboring interface is set to **trunk** or **desirable**.

`Switch(config-if)#switchport nonegotiate`	Prevents the interface from generating DTP frames.
	NOTE: Use the **switchport mode nonegotiate** command only when the interface switchport mode is **access** or **trunk**. You must manually configure the neighboring interface to establish a trunk link.
`Switch(config-if)#switchport mode trunk`	Puts the interface into permanent trunking mode and negotiates to convert the link into a trunk link.
	NOTE: With the **switchport mode trunk** command set, the interface becomes a trunk link even if the neighboring interface is not a trunk link.

TIP: The default mode is dependent on the platform. For the 2960, 3560, and the 3760, the default mode is dynamic auto.

Setting the Trunk Encapsulation and Allowed VLANs

`3560Switch(config)#interface fastethernet 0/1`	Moves to interface configuration mode.
`3560Switch(config-if)# switchport mode trunk`	Puts the interface into permanent trunking mode and negotiates to convert the link into a trunk link.
`3560Switch(config-if)# switchport trunk encapsulation isl`	Specifies Inter-Switch Link (ISL) encapsulation on the trunk link.
`3560Switch(config-if)# switchport trunk encapsulation dot1q`	Specifies 802.1Q encapsulation on the trunk link.
`3560Switch(config-if)# switchport trunk encapsulation negotiate`	Specifies that the interface negotiate with the neighboring interface to become either an ISL or Dot1Q trunk, depending on the capabilities or configuration of the neighboring interface.
`3560Switch(config-if)# switchport trunk allowed vlan 10,12,18-22`	Configures the list of VLANs allowed on the trunk.
	NOTE: All VLANs are allowed by default.
`3560Switch(config-if)# switchport trunk allowed vlan add 44,47-49`	Configures the list of VLANs to add to the existing VLANs allowed on the trunk.

`3560Switch(config-if)#` **`switchport trunk allowed vlan remove 44,47-49`**	Configures the list of VLANs to remove from the existing VLANs allowed on the trunk.
	NOTE: Do not enter any spaces between comma-separated VLAN parameters or in hyphen-specified ranges.

TIP: With the **switchport trunk encapsulation negotiate** command set, the preferred trunking method is ISL.

CAUTION: The 2960 series switch supports only 802.1Q trunking, and therefore the **switchport trunk encapsulation** command is not required.

Verifying VLAN Information

`Switch#`**`show vlan`**	Displays VLAN information.
`Switch#`**`show vlan brief`**	Displays VLAN information in brief.
`Switch#`**`show vlan id 2`**	Displays information of VLAN 2 only.
`Switch#`**`show vlan name marketing`**	Displays information of VLAN named marketing only.
`Switch#`**`show interfaces trunk`**	Display trunk ports, trunking modes, encapsulation, native and allowed VLANs.
`Switch#`**`show interfaces switchport`**	Display administrative and operational status of trunks, encapsulation, private VLAN, voice VLAN, and trunk VLAN pruning.
	NOTE: The preceding two commands can be qualified to show the output for a single interface (for example, **show interface FastEthernet 0/5 trunk**).

Saving VLAN Configurations

The stored configurations of VLANs 1 through 1005 are always saved in the VLAN database, the vlan.dat file in flash:. After creating or deleting a VLAN in VLAN configuration mode, the **exit** command will apply any new changes to the VLAN database.

If you are using VTP transparent mode, the configurations are also saved in the running configuration, and can be saved to the startup configuration using the **copy running-config startup-config** command.

If the VTP mode is transparent in the startup configuration, and the VLAN database and the VTP domain name from the VLAN database matches that in the startup configuration file, the VLAN database is ignored (cleared), and the VTP and VLAN configurations in the startup configuration file are used. The VLAN database revision number remains unchanged in the VLAN database.

Erasing VLAN Configurations

`Switch#delete flash:vlan.dat`	Removes entire VLAN database from flash.
	CAUTION: Make sure that there is *no* space between the colon (:) and the characters *vlan.dat*. You can potentially erase the entire contents of the flash with this command if the syntax is not correct. Make sure to read the output from the switch. If you need to cancel, press Ctrl+C to escape back to privileged mode: Switch# Switch#**delete flash:vlan.dat** Delete filename [vlan.dat]? Delete flash:vlan.dat? [confirm] Switch#
`Switch(config)#interface fastethernet 0/5`	Moves to interface configuration mode.
`Switch(config-if)#no switchport access vlan 5`	Removes port from VLAN 5 and reassigns it to VLAN 1 (the default VLAN).
`Switch(config-if)#exit`	Moves to global configuration mode.
`Switch(config)#no vlan 5`	Removes VLAN 5 from the VLAN database.

NOTE: When you delete a VLAN from a switch that is in VTP server mode, the VLAN is removed from the VLAN database for all switches in the VTP domain. When you delete a VLAN from a switch that is in VTP transparent mode, the VLAN is deleted only on that specific switch.

NOTE: You cannot delete the default VLANs for the different media types: Ethernet VLAN 1 and FDDI or Token Ring VLANs 1002 to 1005.

CAUTION: When you delete a VLAN, any ports assigned to that VLAN become inactive. This "inactive" state can be seen using the **show interface switchport** command for the port or ports in question. The ports remain associated with the VLAN (and thus inactive) until you assign those ports to a defined VLAN. Therefore, it is recommended that you reassign ports to a new VLAN or the default VLAN before you delete a VLAN from the VLAN database.

Verifying VLAN Trunking

`Switch#show interface fastethernet 0/1 trunk`	Displays the administrative and operational status of a trunking port

VLAN Trunking Protocol

VLAN Trunking Protocol (VTP) is a Cisco proprietary protocol that allows for VLAN configuration (addition, deletion, or renaming of VLANs) to be consistently maintained across a common administrative domain. The three versions of VTP (1, 2, and 3) are not interoperable.

One new feature supported in VTP Version 3 is that of a primary and secondary VTP server. The primary VTP server updates the VLAN database for the VTP domain. The secondary VTP server role is to back up to NVRAM the updated VTP configurations from the primary server.

As early as 2007, there is no specific recommendation on whether to use VTP client/server modes or VTP transparent mode.

CAUTION: You should take great care with VLAN changes on the VTP server or the provisioning of switches to be added to an existing VTP domain. An unintentional configuration change at the VTP server or adding a switch with a higher VTP revision level can rewrite the VLAN information in every switch in the VTP domain.

Using Global Configuration Mode

`Switch(config)#vtp domain` *domain-name*	Configures the VTP domain name. The name can be from 1 to 32 characters long.
	NOTE: The VTP domain name cannot be reset to its null state unless the VLAN database, vlan.dat, is deleted.
`Switch(config)#vtp version 1 | 2 | 3`	Configures the VTP version. The VTP version must be the same on all switches.
`Switch(config)#vtp mode client`	Changes the switch to VTP client mode.
`Switch(config)#vtp mode server`	Changes the switch to VTP server mode.
`Switch(config)#vtp mode transparent`	Changes the switch to VTP transparent mode.
	NOTE: By default, all Catalyst switches are in server mode.
	NOTE: In VTP Version 3, all switches come up as secondary servers. You can specify a primary server for database updates by issuing the **vtp primary-server** takeover command on another switch in the VTP domain.
`Switch(config)#no vtp mode`	Returns the switch to the default VTP server mode.
`Switch(config-if)#no vtp`	Disables VTP on a switch port.
	NOTE: All switches operating in VTP server or client mode must have the same domain name to ensure communication. A switch in transparent mode must belong to the VTP domain to forward the VLAN management domain messages.

`Switch(config)#vtp password` *password*	Configures a VTP password. In Cisco IOS Software Release 12.3 and later, the password is an ASCII string from 1 to 32 characters long. If you are using a Cisco IOS release earlier than 12.3 or Version 3, the password length ranges from 8 to 64 characters long.
	NOTE: To communicate with each other, all switches must have the same VTP password set.
`Switch(config)#vtp pruning`	Enables VTP pruning.
	NOTE: By default, VTP pruning is disabled. For Version 1 and 2, you need to enable VTP pruning on only one switch in VTP server mode. In Version 3, the administrator must enable or disable pruning on each device. VTP pruning applies only to VLANs 1 to 1001 in version 1, 2, and 3.
	NOTE: VTP sessions are not interoperable. All switches must use the same version. The biggest difference between Versions 2 and 3 is that Version 3 supports enhanced authentication, extended VLANs, and private VLANs.

NOTE: Only VLANs included in the pruning-eligible list can be pruned. VLANs 2 through 1001 are pruning eligible by default on trunk ports. Reserved VLANs and extended-range VLANs cannot be pruned. To change which eligible VLANs can be pruned, use the interface-specific **switchport trunk pruning vlan** command:

```
Switch(config-if)#switchport trunk pruning vlan remove 4, 20-30
! Removes VLANs 4 and 20-30
Switch(config-if)#switchport trunk pruning vlan except 40-50
! All VLANs are added to the pruning list except for 40-50
```

NOTE: New in VTP Version 3 is the primary server. Only the primary server can make changes to the VLAN database. The primary VTP server is a "run-time" enablement. There is no persistent VTP primary server configuration stored in NVRAM and thus any switch in the domain can be primary server. This persists until the switch is reloaded or another switch in the VTP domain is configured as the VTP primary server.

`Switch(config)#vtp mode off`	Disables VTP messaging on all trunks on the switch.
	NOTE: You can specify VTP on or off on a per-VTP instance basis.
`Switch(config)#vtp domain SW-GRP14`	Configures the VTP administrative domain name.
`Switch(config)#vtp primary-server`	Change the VTP role of a switch from the default secondary server to primary server and advertise the configuration to the domain.

Verifying VTP

`Switch#show vtp status`	Displays general information about VTP configuration
`Switch#show vtp counters`	Displays the VTP counters for the switch

NOTE: If trunking has been established before VTP is set up, VTP information is propagated throughout the switch fabric almost immediately. However, because VTP information is advertised only every 300 seconds (5 minutes) unless a change has been made to force an update, it can take several minutes for VTP information to be propagated.

Configuration Example: VLANs

Figure 9-1 shows the network topology for the configuration that follows, which shows how to configure VLANs using the commands covered in this chapter.

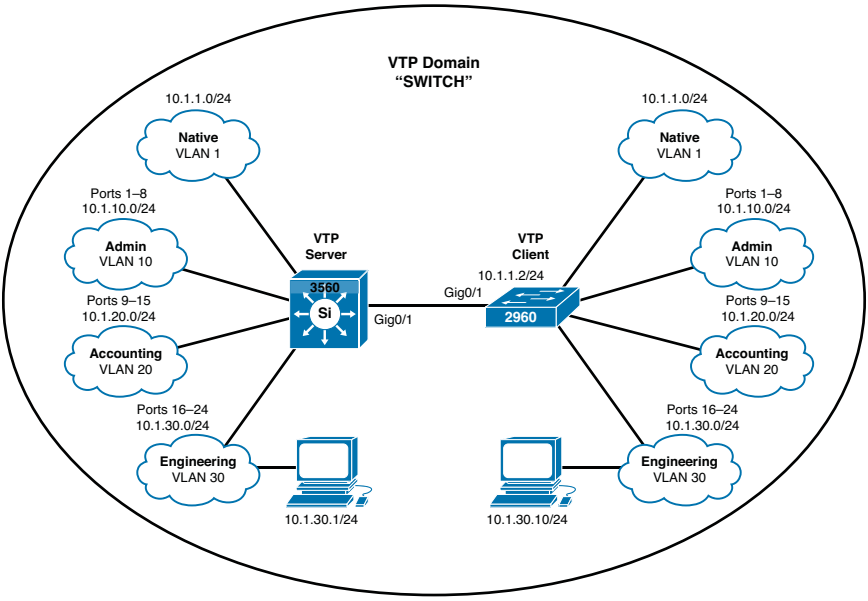


Figure 9-1 Network Topology for VLAN Configuration Example

3560 Switch

`Switch> enable`	Moves to privileged mode.
`Switch#configure terminal`	Moves to global configuration mode.
`Switch(config)#hostname 3560`	Sets the hostname.

`3560(config)#vtp mode server`	Changes the switch to VTP server mode. Note that server is the default setting for a 3560 switch.
`3560(config)#vtp domain SWITCH`	Configures the VTP domain name to SWITCH.
`3560(config)#vtp password Order66`	Sets the VTP password to **Order66**.
`3560(config)#vlan 10`	Creates VLAN 10 and enters VLAN configuration mode.
`3560(config-vlan)#name Admin`	Assigns a name to the VLAN.
`3560(config-vlan)#exit`	Increases the revision number by 1 and returns to global configuration mode.
`3560(config)#vlan 20`	Creates VLAN 20 and enters VLAN configuration mode.
`3560(config-vlan)#name Accounting`	Assigns a name to the VLAN.
`3560(config-vlan)#vlan 30`	Creates VLAN 30 and enters VLAN configuration mode. You do not have to exit back to global configuration mode to execute this command.
	NOTE: The VTP revision number would be incremented.
`3560(config-vlan)#name Engineering`	Assigns a name to the VLAN.
`3560(config-vlan)#exit`	Increases the revision number by 1 and returns to global configuration mode.
`3560(config)#interface range fasthethernet 0/1 - 8`	Enables you to set the same configuration parameters on multiple ports at the same time.
`3560(config-if-range)#switchport mode access`	Sets ports 1–8 as access ports.
`3560(config-if-range)#switchport access vlan 10`	Assigns ports 1–8 to VLAN 10.
`3560(config-if-range)#interface range fastethernet 0/9 - 15`	Enables you to set the same configuration parameters on multiple ports at the same time.
`3560(config-if-range)#switchport mode access`	Sets ports 9–15 as access ports.
`3560(config-if-range)#switchport access vlan 20`	Assigns ports 9–15 to VLAN 20.
`3560(config-if-range)#interface range fastethernet 0/16 - 24`	Enables you to set the same configuration parameters on multiple ports at the same time.
`3560(config-if-range)#switchport mode access`	Sets ports 16–24 as access ports.

`3560(config-if-range)#switchport access vlan 30`	Assigns ports 16–24 to VLAN 30.
`3560(config-if-range)#exit`	Returns to global configuration mode.
`3560(config)#interface gigabitethernet 0/1`	Moves to interface configuration mode.
`3560(config-if)#switchport trunk encapsulation dot1q`	Specifies 802.1Q encapsulation on the trunk link.
`3560(config-if)#switchport mode trunk`	Puts the interface into permanent trunking mode and negotiates to convert the link into a trunk link.
`3560(config-if)#exit`	Returns to global configuration mode.
`3560(config)#vtp version 3`	Enables VTP Version 3.
`3560(config)#vtp pruning`	Enables VTP pruning on this switch.
`3560(config)#exit`	Returns to privileged mode.
`3560#copy running-config startup-config`	Saves the configuration in NVRAM.

2960 Switch

`Switch>enable`	Moves to privileged mode
`Switch#configure terminal`	Moves to global configuration mode
`Switch(config)#hostname 2960`	Sets the hostname
`2960(config)#vtp mode client`	Changes the switch to VTP server mode
`2960(config)#vtp domain Order66`	Configures the VTP domain name to **Order66**
`2960(config)#interface range fastethernet 0/1 - 8`	Enables you to set the same configuration parameters on multiple ports at the same time
`2960(config-if-range)#switchport mode access`	Sets ports 1–8 as access ports
`2960(config-if-range)#switchport access vlan 10`	Assigns ports 1–8 to VLAN 10
`2960(config-if-range)#interface range fastethernet 0/9 - 15`	Enables you to set the same configuration parameters on multiple ports at the same time
`2960(config-if-range)#switchport mode access`	Sets ports 9–15 as access ports
`2960(config-if-range)#switchport access vlan 20`	Assigns ports 9–15 to VLAN 20
`2960(config-if-range)#interface range fastethernet 0/16 - 24`	Enables you to set the same configuration parameters on multiple ports at the same time

`2960(config-if-range)#switchport mode access`	Sets ports 16–24 as access ports
`2960(config-if-range)#switchport access vlan 30`	Assigns ports 16–24 to VLAN 30
`2960(config-if-range)#exit`	Returns to global configuration mode
`2960(config)#int gigabitethernet 0/1`	Moves to interface configuration mode
`2960(config-if)#switchport mode trunk`	Puts the interface into permanent trunking mode and negotiates to convert the link into a trunk link
`2960(config-if)#exit`	Returns to global configuration mode
`2960(config)#vtp version 3`	Enables VTP Version 3 on this switch
`2960(config)#vtp pruning`	Enables VTP pruning on this switch
`2960(config)#exit`	Returns to privileged mode
`2960#copy running-config startup-config`	Saves the configuration in NVRAM

3560 Switch

`3560>enable`	Moves to privileged mode
`3560#configure terminal`	Moves to global configuration mode
`3560(config)#vlan 999`	Creates VLAN 999 (unsuccessful)
	NOTE: VTP VLAN configuration not allowed when device is not the primary server for VLAN database.
`3560#vtp primary-server`	Configures the 3560 to be the VTP primary server
`3560(config)#vlan 999`	Creates VLAN 999 (successful)
`3560(config-vlan)#exit`	Returns to global configuration mode
`3560(config)#end`	Returns to privileged mode
`3560#copy running-config startup-config`	Saves the configuration in NVRAM

Layer 2 Link Aggregation

EtherChannel provides fault-tolerant high-speed links between switches, routers, and servers. An EtherChannel consists of individual Fast Ethernet or Gigabit Ethernet links bundled into a single logical link. If a link within an EtherChannel fails, traffic previously carried over that failed link changes to the remaining links within the EtherChannel.

Link Aggregation Interface Modes

Mode	Protocol	Description
On	None	Forces the interface into aggregation without Port Aggregation Protocol (PAgP) or Link Aggregation Control Protocol (LACP). Channel only exists if connected to another interface group also in on mode.
Auto	PAgP (Cisco)	Places the interface into a passive negotiating state (will respond to PAgP packets, but will not initiate PAgP negotiation).
Desirable	PAgP (Cisco)	Places the interface into an active negotiating state (will send PAgP packets to start negotiations).
Passive	LACP (IEEE)	Places the interface into a passive negotiating state (will respond to LACP packets, but will not initiate LACP negotiation).
Active	LACP (IEEE)	Places the interface into an active negotiating state (will send LACP packets to start negotiations).

Guidelines for Configuring Link Aggregation

- PAgP is Cisco proprietary and not compatible with LACP.
- LACP is defined in 802.3ad.
- Can combine from two to eight parallel links.
- All ports must be identical:
  - Same speed and duplex
  - Cannot mix Fast Ethernet and Gigabit Ethernet
  - Cannot mix PAgP and LACP
  - Must all be VLAN trunk or nontrunk operational status
  - All VLANs and allowed VLANs must match
- All links must be either L2 or L3 in a single channel group.
- To create a channel in PAgP, sides must be set to
  - Auto-
  - Desirable-
- To create a channel in LACP, sides must be set to
  - Active
  - Passive
- To create a channel without using PAgP or LACP, sides must be set to on-on.
- Do *not* configure a GigaStack gigabit interface converter (GBIC) as part of an EtherChannel.
- An interface that is already configured to be a Switched Port Analyzer (SPAN) destination port will not join an EtherChannel group until SPAN is disabled.

- Do *not* configure a secure port as part of an EtherChannel.
- Interfaces with different native VLANs cannot form an EtherChannel.
- When using trunk links, ensure all trunks are in the same mode: Inter-Switch Link (ISL) or Dot1Q.

Configuring L2 EtherChannel

`Switch(config)#interface port-channel {number}` `Switch(config-if)#interface parameters`	Specifies the port channel interface. Once in the interface configuration mode, you can configure additional parameters.
`Switch(config)#interface range fastethernet 0/1 - 4`	Moves to interface range configuration mode.
`3560Switch(config-if-range)#channel-group 1 mode on`	Creates channel group 1 as an EtherChannel and assigns interfaces 20–24 as part of it.
`3560Switch(config-if-range)#channel-group 1 mode desirable`	Creates channel group 1 as an PAgP channel and assigns interfaces 20–24 as part of it.
`3560Switch(config-if-range)#channel-group 1 mode active`	Creates channel group 1 as a LACP channel and assigns interfaces 20–24 as part of it.

Configuring L3 EtherChannel

`3560Switch(config)#interface port-channel 1`	Creates the port channel logical interface, and moves to interface config mode. Valid channel numbers are 1–48.
`3560Switch(config-if)#no switchport`	Puts the interface into Layer 3 mode.
`3560Switch(config-if)#ip address 172.16.10.1 255.255.255.0`	Assigns IP address and netmask.
`3560Switch(config-if)#exit`	Moves to global configuration mode.
`3560Switch(config)#interface range fastethernet 0/20 - 24`	Moves to interface range configuration mode.
`3560Switch(config-if-range)#no ip address`	Ensures there are no IP addresses assigned on the interfaces.
`3560Switch(config-if-range)#channel-group 1 mode on`	Creates channel group 1 as an EtherChannel and assigns interfaces 20–24 as part of it.
`3560Switch(config-if-range)#channel-group 1 mode desirable`	Creates channel group 1 as an PAgP channel and assigns interfaces 20–24 as part of it.

`3560Switch(config-if-range)#` **`channel-group 1 mode active`**	Creates channel group 1 as a LACP channel and assigns interfaces 20–24 as part of it.
	NOTE: The channel group number must match the port channel number.

Verifying EtherChannel

`Switch#`**`show running-config`**	Displays list of what is currently running on the device.
`Switch#`**`show running-config interface fastethernet 0/12`**	Displays interface FastEthernet0/12 information
`Switch#`**`show interfaces fastethernet 0/12 etherchannel`**	Displays L3 EtherChannel information
`Switch#`**`show etherchannel`**	Displays all EtherChannel information
`Switch#`**`show etherchannel 1 port-channel`**	Displays port channel information
`Switch#`**`show etherchannel summary`**	Displays a summary of EtherChannel information
`Switch#`**`show pagp neighbor`**	Shows PAgP neighbor information
`Switch#`**`clear pagp 1 counters`**	Clears PAgP channel group 1 information
`Switch#`**`clear lacp 1 counters`**	Clears LACP channel group 1 information

Configuring EtherChannel Load Balancing

`Switch(config)#`**`port-channel load-balance`** *`type`*	Configures load balancing of method named *type*.
	NOTE: The following methods are allowed when load balancing across a port channel: **dst-ip:** Distribution is based on destination host IP address. **dst-mac:** Distribution is based on the destination MAC address. Packets to the same destination are sent on the same port, but packets to different destinations are sent on different ports in the channel. **src-dst-ip:** Distribution is based on source and destination host IP address. **src-dst-mac:** Distribution is based on source and destination MAC address. **src-ip:** Distribution is based on source IP address.**src-mac:** Distribution is based on source MAC address. Packets from different hosts use different ports in the channel, but packets from the same host use the same port.
`Switch#`**`show etherchannel load-balance`**	Displays EtherChannel load-balancing information.

Configuration Example: PAgP EtherChannel

Figure 9-2 shows the network topology for the configuration that follows, which shows how to configure EtherChannel using commands covered in this chapter.

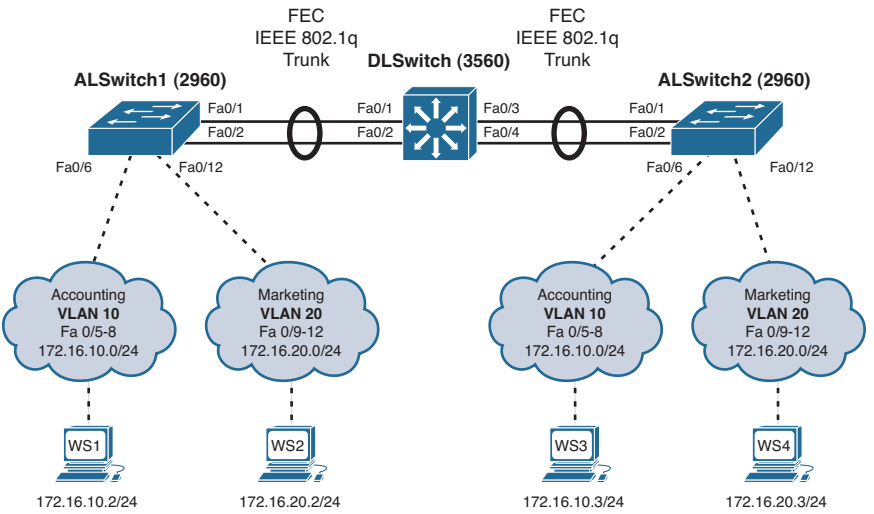


Figure 9-2 Network Topology for EtherChannel Configuration

DLSwitch (3560)

`Switch> enable`	Moves to privileged mode
`Switch#configure terminal`	Moves to global configuration mode
`Switch(config)#hostname DLSwitch`	Sets hostname
`DLSwitch(config)#no ip domain-lookup`	Turns off DNS queries so that spelling mistakes will not slow you down
`DLSwitch(config)#vtp mode server`	Changes the switch to VTP server mode
`DLSwitch(config)#vtp domain testdomain`	Configures the VTP domain name to testdomain
`DLSwitch(config)#vlan 10`	Creates VLAN 10 and enters VLAN configuration mode.
`DLSwitch(config-vlan)#name Accounting`	Assigns a name to the VLAN
`DLSwitch(config-vlan)#exit`	Returns to global configuration mode
`DLSwitch(config)#vlan 20`	Creates VLAN 20 and enters VLAN configuration mode
`DLSwitch(config-vlan)#name Marketing`	Assigns a name to the VLAN
`DLSwitch(config-vlan)#exit`	Returns to global configuration mode

DLSwitch(config)#**interface range fastethernet 0/1 - 4**	Moves to interface range configuration mode
DLSwitch(config-if)#**switchport trunk encapsulation dot1q**	Specifies 802.1Q encapsulation on the trunk link
DLSwitch(config-if)#**switchport mode trunk**	Puts the interface into permanent trunking mode and negotiates to convert the link into a trunk link
DLSwitch(config-if)#**exit**	Returns to global configuration mode
DLSwitch(config)#**interface range fastethernet 0/1 - 2**	Moves to interface range configuration mode
DLSwitch(config-if)#**channel-group 1 mode desirable**	Creates channel group 1 and assigns interfaces 0/1–0/2 as part of it
DLSwitch(config-if)#**exit**	Moves to global configuration mode
DLSwitch(config)#**interface range fastethernet 0/3 - 4**	Moves to interface range configuration mode
DLSwitch(config-if)#**channel-group 2 mode desirable**	Creates channel group 2 and assigns interfaces 03–04 as part of it
DLSwitch(config-if)#**exit**	Moves to global configuration mode
DLSwitch(config)#**port-channel load-balance dst-mac**	Configures load balancing based on destination MAC address
DLSwitch(config)#**exit**	Moves to privileged mode
DLSwitch#**copy running-config startup-config**	Saves the configuration to NVRAM

ALSwitch1 (2960)

Switch>**enable**	Moves to privileged mode
Switch#**configure terminal**	Moves to global configuration mode
Switch(config)#**hostname ALSwitch1**	Sets hostname
ALSwitch1(config)#**no ip domain-lookup**	Turns off DNS queries so that spelling mistakes will not slow you down
ALSwitch1(config)#**vtp mode client**	Changes the switch to VTP client mode
ALSwitch1(config)#**vtp domain testdomain**	Configures the VTP domain name to testdomain
ALSwitch1(config)#**interface range fastethernet 0/5 - 8**	Moves to interface range configuration mode
ALSwitch1(config-if-range)# **switchport mode access**	Sets ports 5–8 as access ports
ALSwitch1(config-if-range)# **switchport access vlan 10**	Assigns ports to VLAN 10
ALSwitch1(config-if-range)#**exit**	Moves to global configuration mode

ALSwitch1(config)#**interface range fastethernet 0/9 - 12**	Moves to interface range configuration mode
ALSwitch1(config-if-range)# **switchport mode access**	Sets ports 9–12 as access ports
ALSwitch1(config-if-range)# **switchport access vlan 20**	Assigns ports to VLAN 20
ALSwitch1(config-if-range)#**exit**	Moves to global configuration mode
ALSwitch1(config)#**interface range fastethernet 0/1 - 2**	Moves to interface range configuration mode
ALSwitch1(config-if-range)# **switchport mode trunk**	Puts the interface into permanent trunking mode and negotiates to convert the link into a trunk link
ALSwitch1(config-if-range)# **channel-group 1 mode desirable**	Creates channel group 1 and assigns interfaces 0/1–0/2 as part of it
ALSwitch1(config-if-range)#**exit**	Moves to global configuration mode
ALSwitch1(config)#**exit**	Moves to privileged mode
ALSwitch1#**copy running-config startup-config**	Saves the configuration to NVRAM

ALSwitch2 (2960)

Switch> **enable**	Moves to privileged mode.
Switch#**configure terminal**	Moves to global config mode.
Switch(config)#**hostname ALSwitch2**	Sets hostname.
ALSwitch2(config)#**no ip domain-lookup**	Turns off DNS queries so that spelling mistakes will not slow you down.
ALSwitch2(config)#**vtp mode client**	Changes the switch to VTP client mode.
ALSwitch2(config)#**vtp domain testdomain**	Configures the VTP domain name to testdomain.
ALSwitch2(config)#**interface range fastethernet 0/5 - 8**	Moves to interface range configuration mode.
ALSwitch2(config-if-range)# **switchport mode access**	Sets ports 5–8 as access ports.
ALSwitch2(config-if-range)# **switchport access vlan 10**	Assigns ports to VLAN 10.
ALSwitch2(config-if-range)#**exit**	Moves to global configuration mode.
ALSwitch2(config)#**interface range fastethernet 0/9 - 12**	Moves to interface range configuration mode.
ALSwitch2(config-if-range)# **switchport mode access**	Sets ports 9–12 as access ports.
ALSwitch2(config-if-range)# **switchport access vlan 20**	Assigns ports to VLAN 20.

`ALSwitch2(config-if-range)#exit`	Moves to global configuration mode.
`ALSwitch2(config)#interface range fastethernet 0/1 - 2`	Moves to interface range configuration mode.
`ALSwitch2(config-if-range)#switchport mode trunk`	Puts the interface into permanent trunking mode and negotiates to convert the link into a trunk link.
`ALSwitch2(config-if-range)#channel-group 2 mode desirable`	Creates channel group 2 and assigns interfaces 01–02 as part of it.
	NOTE: Although the local channel group number does not have to match the channel group number on a neighboring switch, the numbers are often chosen to be the same for ease of management and documentation purposes.
`ALSwitch2(config-if-range)#exit`	Moves to global configuration mode.
`ALSwitch2(config)#exit`	Moves to privileged mode.
`ALSwitch2#copy running-config startup-config`	Saves the configuration to NVRAM.

DHCP for IPv4

Configuring Basic DHCP Server for IPv4

`Switch(config)#ip dhcp excluded-address 172.22.12.1 172.22.12.31`	Selects the range of IP address that will not be assigned by the DHCP service
`Switch(config)#ip dhcp pool VLAN18_POOL1`	Creates a DHCP pool named VLAN18_POOL1
`Switch(dhcp-config)#network 172.22.12.0 /24`	Defines the IP network for the pool in dotted decimal with subnet mask or CIDR notation
`Switch(dhcp-config)#default-router 172.22.12.1`	Specifies the gateway router for the DHCP clients
`Switch(dhcp-config)#dns-server 192.168.22.11`	Specifies the IP of the DNS service
`Switch(dhcp-config)#lease 1 0 0`	Specifies the DHCP lease length in “days hours minutes”
`Switch(dhcp-config)#exit`	Leave DHCP configuration mode

Configuring DHCP Manual IP Assignment for IPv4

It is sometimes desirable to link a specific network device with a specific IPv4 address using the switch's DHCP service. The switch uses a "client ID" to identify a DHCP client device and is programmed into the DHCP pool.

NOTE: The DHCP client device ID can be determined using the **show ip dhcp binding** command after the client has successfully obtained the next available IP address from the DHCP pool.

The DHCP pool programming must also include any other required programming such as default router IP, DNS or WINS addresses, and so on.

`Switch(config)#ip dhcp pool POOL1`	Creates a DHCP pool named POOL1
`Switch(dhcp-config)#host 172.22.12.88 /24`	Defines the single IP address for the DHCP pool in dotted decimal with subnet mask or CIDR notation
`Switch(dhcp-config)#client-identifier client-identifier 0063.6973.636f.2d30.3030.362e.6636.3962.2e65.3331.312d.4769.302f.31`	Specifies the client ID of the network device that should receive the specific IP
`Switch(dhcp-config)#default-router 172.22.12.1`	Specifies the gateway router for the DHCP clients
`Switch(dhcp-config)#dns-server 192.168.22.11`	Specifies the IP of the DNS service
`Switch(dhcp-config)#lease 1 0 0`	Specifies the DHCP lease length in "days hours minutes"
`Switch(dhcp-config)#exit`	Leaves DHCP configuration mode

Implementing DHCP Relay IPv4

NOTE: DHCP services can reside anywhere within the network. The DHCP relay service translates a client broadcast DHCP service request to a unicast DHCP request directed to the DHCP server IP. The command is added to the Layer 3 interface on the IP segment from which the DHCP broadcast request originates. The **ip helper-address interface** command forwards eight UDP services by default. They are by service and port:

Time/37

TACACS/49

DNS/53

BOOTP-DHCP Server/67

BOOTP-DHCP Client/68

TFTP/69

NetBIOS name service/137NetBIOS datagram service/138

Services not forwarded by **ip helper-address** can be added using the **ip forward-protocol** global command.

`Switch(config-if)#ip helper-address 10.1.1.1`	Forward the DCHP traffic to the DHCP server at 10.1.1.1
`Switch(config)#no ip forward-protocol udp 37`	Do not forward traffic for UDP time services using port 37
`Switch(config)#ip forward-protocol udp 5858`	Forward traffic for UDP services using port 5858

Verifying DHCP for IPv4

`Switch#show ip dhcp binding`	Display the IPv4 to MAC address bindings
`Switch#show ip dhcp pool`	Displays DHCPv4 pool statistics
`Switch #show ip dhcp interface`	Displays interface on which DHCPv4 is enabled
`Switch #debug ip dhcp server events`	Report address assignments, lease expirations, and so on
`Switch #debug ip dhcp server packets`	Decode DHCP server message receptions and transmissions

Implementing DHCP for IPv6

DHCPv6 can deliver both stateful and stateless information. Stateful, or centrally managed, information is used to provide parameters not available through autoconfig or neighbor discovery. Stateless address autoconfiguration (SLAAC) means that the client picks their own address based on the router prefix being advertised. Additional parameters such as a DNS server address must be provided by the DHCPv6 service.

The DHCPv6 prefix delegation option can automate the assignment of CPE customer devices from provider-edge devices.

DHCPv6 clients and servers are identified to each other by a DHCP unique identifier (DUID) using the lowest number interface MAC address. DHCPv6 exchanges are either normal four-message (solicit, advertise, request, reply) or the rapid commit two-message (solicit, reply).

The DHCPv6 server maintains a binding table in RAM that maintains configuration parameters.

NOTE: Unlike DHCPv4, the DHCPv6 service does not give out IP addresses; instead, it gives out prefixes. The client creates the remaining bits for a valid IPv6 address. The duplicate address detection mechanism ensures the uniqueness of the address. There is no DHCPv6 **excluded-address** command.

Configuring DHCPv6 Server

`Switch#configure terminal`	Enters global configuration mode
`Switch(config)#ip routing`	Enables the switch's Layer 3 functions
`Switch(config)#sdm prefer dual-ipv4-and-ipv6 routing`	Configures TCAM and forwarding RAM sizes to facilitate IPv6 functions
`Switch(config)#ipv6 dhcp pool POOL1`	Creates a DHCPv6 pool named POOL1
`Switch(config-dhcpv6)#address prefix 2001:db8:14::/64 lifetime infinite infinite`	Specifies an address prefix for address assignment, including an optional address lifetime parameter
`Switch(config-dhcp)#domain-name nodomain.com`	Configures a domain name for a DHCPv6 client
`Switch(config-dhcp)#dns-server 2001:DB8:3000:3000::42`	Specifies the DNS server address for the DHCPv6 clients
`Switch(config-dhcp)#exit`	Leaves DHCPv6 configuration mode
`Switch(config)#interface vlan 21`	Specifies an interface type and number, and enters interface configuration mode
`Switch(config)#ipv6 address 2001:db8:14::1/64`	Assigns an IPv6 address to the interface
`Switch(config-if)#ipv6 dhcp server POOL1`	Enables DHCPv6 on an interface for the appropriate IPv6 address pool
`Switch(config-if)#end`	Moves to privilege EXEC mode

Configuring DHCPv6 Client

`Switch#configure terminal`	Enters global configuration mode
`Switch#configure terminal`	Enters global configuration mode
`Switch(config)#interface` *`interface-id`*	Enters interface configuration mode, and specify the interface to configure
`Switch(config-if)#ipv6 address dhcp`	Enables the interface to acquire an IPv6 address using the four-message exchange from the DHCPv6 server
`Switch(config-if)#ipv6 address dhcp rapid-commit`	Enables the interface to acquire an IPv6 address using the two-message exchange from the DHCPv6 server

Configuring DHCPv6 Relay Agent

`Switch#configure terminal`	Enables privileged EXEC mode
`Switch(config)#interface ethernet 4/2`	Specifies an interface type and number, and enters interface configuration mode
`Switch (config-if) ipv6 dhcp relay destination FE80::250:A2FF:FEBF:A056 ethernet 4/3`	Specifies a destination address to which client packets are forwarded and enables DHCPv6 relay service on the interface
`Switch(config-if)#end`	Return to privileged EXEC mode

Verifying DHCPv6

`Switch#show ipv6 dhcp binding`	Display the IPv6 to MAC address bindings
`Switch#show ipv6 dhcp pool`	Displays DHCPv6 pool statistics
`Switch #show ipv6 dhcp interface`	Displays interface on which DHCPv6 is enabled
`Switch #debug ipv6 dhcp [detail]`	Enables DHCPv6 debugging
`Switch #debug ipv6 dhcp relay`	Enables DHCPv6 relay agent debugging

CHAPTER 10

Implementing Spanning Tree

This chapter provides information about the following topics:

- Spanning-Tree Standards
  - Enabling Spanning Tree Protocol
  - Configuring the root switch
  - Configuring a secondary root switch
  - Configuring port priority
  - Configuring the path cost
  - Configuring the switch priority of a VLAN
  - Configuring STP timers
  - Verifying STP
  - Cisco STP Toolkit
    - PortFast
    - BPDU Guard
    - BPDU Filter
    - UplinkFast
    - BackboneFast
    - Root Guard
    - Loop Guard
    - Unidirectional link detection
  - Port error conditions
  - FlexLinks
  - Changing the spanning-tree mode
  - Extended system ID
  - Enabling Rapid Spanning Tree
  - Enabling Multiple Spanning Tree
  - Verifying MST
  - Troubleshooting STP
  - Configuration example: PVST+
  - Spanning-Tree migration example: PVST+ to Rapid-PVST+

CAUTION: Your hardware platform or software release might not support all the commands documented in this chapter. Please refer to the Cisco website for specific platform and software release notes.

Spanning-Tree Standards

The spanning tree standards provide the same safety that routing protocols provide in Layer 3 forwarding environments to Layer 2 bridging environments. A single best path to a main bridge is found and maintained in the Layer 2 domain, and other redundant paths are managed by selective port blocking. Appropriate blocked ports begin forwarding when primary path(s) to the main bridge are no longer available.

The IEEE published the first Spanning Tree Protocol (STP) standard, 802.1D, in 1990. The last version of 802.1D was published in 2004 and included a number of enhancements. The 802.1D standard supported a single common spanning tree.

In 2001 the IEEE published the Rapid Spanning Tree Protocol (RSTP) standard, 802.1w. This standard relied less on state machine timers and more on "loop protecting" real-time switch-to-switch negotiation after a topology change. The selection of ports for blocking or forwarding was fast as was the flushing of invalid MAC addresses in the affected switches. The 802.1w standard, like the 802.1D standard, supported a single common spanning-tree instance.

Multiple Instance Spanning Tree Protocol (MISTP), IEEE 802.1s, allows several VLANs to be mapped to a reduced number of spanning-tree instances. Cisco curriculums refer to IEEE 802.1s as Multiple Spanning Tree (MST). Each MST instance handles multiple VLANs that have the same Layer 2 topology.

NOTE: Enabling MST enables RSTP.

There are two Cisco proprietary STPs in common use: Per VLAN Spanning Tree Plus (PVST+) and Per VLAN Rapid Spanning Tree Plus (PVRST+). Both protocols allow an instance of either STP or RSTP to run on each VLAN configured on the switch. PVST+ is based on the IEEE 802.1D standard and includes Cisco proprietary extensions such as BackboneFast, UplinkFast, and PortFast. PVRST+ is based on the IEEE 802.1w standard and has a faster convergence than 802.1D.

NOTE: Default spanning-tree implementation for Catalyst 2950, 2960, 3550, 3560, 3750 switches is PVST+. This is a per-VLAN implementation of 802.1D.

Enabling Spanning Tree Protocol

`Switch(config)#spanning-tree vlan 5`	Enables STP on VLAN 5.
`Switch(config)#no spanning-tree vlan 5`	Disables STP on VLAN 5.
	NOTE: Spanning tree is enabled by default on all VLANs.

NOTE: Many access switches such as the Catalyst 2950, 2960, 3550, 3560, 3750 support a maximum 128 spanning trees using any combination of PVST+ or PVRST+. Any VLANs created in excess of 128 will not have a spanning-tree instance running in them. There is a possibility of an L2 loop that could not be broken in the case where a VLAN without spanning tree is transported across a trunk. It is recommended that you use Multiple STP if the number of VLANs in a common topology is high.

Configuring the Root Switch

`Switch(config)#`**`spanning-tree vlan 5 root`**	Modifies the switch priority from the default 32,768 to a lower value to allow the switch to become the root switch for VLAN 5.
	NOTE: This switch resets its priority to 24,576. If any other switch has a priority set to below 24,576 already, this switch sets its own priority to 4096 *less* than the lowest switch priority. If by doing this the switch would have a priority of less than 1, this command fails.
`Switch(config)#`**`spanning-tree vlan 5 root primary`**	Configures the switch to become the root switch for VLAN 5.
	NOTE: The maximum switch topology width and the hello time can be set within this command.
	TIP: The root switch should be a backbone or distribution switch.
`Switch(config)#`**`spanning-tree vlan 5 root primary diameter 7`**	Configures the switch to be the root switch for VLAN 5 and sets the network diameter to 7.
	TIP: The **diameter** keyword is used to define the maximum number of switches between any two end stations. The range is from 2 to 7 switches.
`Switch(config)#`**`spanning-tree vlan 5 root primary hello-time 4`**	Configures the switch to be the root switch for VLAN 5 and sets the hello-delay timer to 4 seconds.
	TIP: The **hello-time** keyword sets the hello-delay timer to any amount between 1 and 10 seconds. The default time is 2 seconds.

Configuring a Secondary Root Switch

`Switch(config)#spanning-tree vlan 5 root secondary`	Configures the switch to become the root switch for VLAN 5 should the primary root switch fail.
	NOTE: This switch resets its priority to 28,672. If the root switch fails, and all other switches are set to the default priority of 32,768, this becomes the new root switch.
`Switch(config)#spanning-tree vlan 5 root secondary diameter 7`	Configures the switch to be the secondary root switch for VLAN 5 and sets the network diameter to 7.
`Switch(config)#spanning-tree vlan 5 root secondary hello-time 4`	Configures the switch to be the secondary root switch for VLAN 5 and sets the hello-delay timer to 4 seconds.

Configuring Port Priority

`Switch(config)#interface gigabitethernet 0/1`	Moves to interface configuration mode.
`Switch(config-if)#spanning-tree port-priority 64`	Configures the port priority for the interface that is an access port.
`Switch(config-if)#spanning-tree vlan 5 port-priority 64`	Configures the VLAN port priority for an interface that is a trunk port.
	NOTE: Port priority is used to break a tie when two switches have equal priorities for determining the root switch. The number can be between 0 and 240 in increments of 16. The default port priority is 128. The lower the number, the higher the priority.

Configuring the Path Cost

`Switch(config)#interface gigabitethernet 0/1`	Moves to interface configuration mode.
`Switch(config-if)#spanning-tree cost 100000`	Configures the cost for the interface that is an access port.
`Switch(config-if)#spanning-tree vlan 5 cost 1000000`	Configures the VLAN cost for an interface that is a trunk port.
	NOTE: If a loop occurs, STP uses the path cost when trying to determine which interface to place into the forwarding state. A higher path cost means a lower speed transmission. The range of the **cost** keyword is 1 through 200,000,000. The default is based on the media speed of the interface.

Configuring the Switch Priority of a VLAN

`Switch(config)#spanning-tree vlan 5 priority 12288`	Configures the switch priority of VLAN 5 to 12288

NOTE: With the **priority** keyword, the range is 0 to 61,440 in increments of 4096. The default is 32,768. The lower the priority, the more likely the switch will be chosen as the root switch.

Only the following numbers can be used as a priority value.

0	4096	8192	12288
16384	20480	24576	28672
32768	36864	40960	45056
49152	53248	57344	61440

CAUTION: Cisco recommends caution when using this command. Cisco further recommends that the **spanning-tree vlan** *x* **root primary** or the **spanning-tree vlan** *x* **root secondary** command be used instead to modify the switch priority.

Configuring STP Timers

`Switch(config)#spanning-tree vlan 5 hello-time 4`	Changes the hello-delay timer to 4 seconds on VLAN 5
`Switch(config)#spanning-tree vlan 5 forward-time 20`	Changes the forward-delay timer to 20 seconds on VLAN 5
`Switch(config)#spanning-tree vlan 5 max-age 25`	Changes the maximum-aging timer to 25 seconds on VLAN 5

NOTE: For the **hello-time** command, the range is 1 to 10 seconds. The default is 2 seconds.

For the **forward-time** command, the range is 4 to 30 seconds. The default is 15 seconds.For the **max-age** command, the range is 6 to 40 seconds. The default is 20 seconds.

CAUTION: Cisco recommends caution when using this command. Cisco further recommends that the **spanning-tree vlan** *x* **root primary** or the **spanning-tree vlan** *x* **root secondary** command be used instead to modify the switch timers.

Verifying STP

`Switch#show spanning-tree`	Displays STP information
`Switch#show spanning-tree active`	Displays STP information on active interfaces only
`Switch#show spanning-tree brief`	Displays a brief status of the STP
`Switch#show spanning-tree detail`	Displays a detailed summary of interface information
`Switch#show spanning-tree interface gigabitethernet 0/1`	Displays STP information for interface GigabitEthernet 0/1
`Switch#show spanning-tree summary`	Displays a summary of port states
`Switch#show spanning-tree summary totals`	Displays the total lines of the STP section
`Switch#show spanning-tree vlan 5`	Displays STP information for VLAN 5

Cisco STP Toolkit

Although the following commands are not mandatory for STP to work, you might find these helpful in fine-tuning your network.

PortFast

`Switch(config)#interface fastethernet 0/10`	Moves to interface configuration mode.
`Switch(config-if)#spanning-tree portfast`	Enables PortFast on an access port.
`Switch(config-if)#spanning-tree portfast trunk`	Enables PortFast on a trunk port.
	CAUTION: Use the **portfast** command only when connecting a single end station to an access or trunk port. Using this command on a port connected to a switch, router, or hub could prevent spanning tree from detecting loops.
	NOTE: If you enable the voice VLAN feature, PortFast is enabled automatically. If you disable voice VLAN, PortFast is still enabled.
`Switch(config)#spanning-tree portfast default`	Globally enables PortFast on all switch ports that are nontrunking.
	NOTE: You can override the **spanning-tree portfast default** global configuration command by using the **spanning-tree portfast** interface configuration command.
`Switch#show spanning-tree interface fastethernet 0/10 portfast`	Displays PortFast information on interface FastEthernet 0/10.

BPDU Guard

`Switch(config)#spanning-tree portfast bpduguard default`	Globally enables BPDU Guard on ports where "PortFast" in enabled.
`Switch(config)#interface range fastethernet 0/1 - 5`	Enters interface range configuration mode.
`Switch(config-if-range)#spanning-tree portfast`	Enables PortFast on all interfaces in the range.
`Switch(config-if-range)#spanning-tree bpduguard enable`	Enables BPDU Guard on all interfaces in the range.
	NOTE: By default, BPDU Guard is disabled.
`Switch(config)#errdisable recovery cause bpduguard`	Allows port to reenable itself if the cause of the error is BPDU Guard by setting a recovery timer.
`Switch(config)#errdisable recovery interval 400`	Sets recovery timer to 400 seconds. Default is 300 seconds. The range is from 30 to 86400 seconds.
`Switch#show spanning-tree summary totals`	Verifies whether BPDU Guard is enabled or disabled.
`Switch#show errdisable recovery`	Displays err-disable recovery timer information.

BPDU Filter

`Switch(config)#spanning-tree portfast bpdufilter default`	Globally enables BPDU filtering on PortFast-enabled port; prevents ports in PortFast from sending or receiving bridge protocol data units (BPDUs).
`Switch(config)#interface range fastethernet 0/1 - 5`	Enters interface range configuration mode.
`Switch(config-if-range)#spanning-tree portfast`	Enables PortFast on all interfaces in the range.
`Switch(config-if-range)#spanning-tree bpdufilter enable`	Enables BPDU Filter on all interfaces in the range configured with "PortFast."
	NOTE: By default, BPDU filtering is disabled.
	CAUTION: Enabling BPDU filtering on an interface, or globally, is the same as disabling STP, which can result in spanning-tree loops being created but not detected.
`Switch#show spanning-tree summary totals`	Displays global BPDU filtering configuration information.
`Switch#show spanning-tree interface [interface-type, interface-number] detail`	Displays detailed spanning-tree interface status and configuration information of the specified interface.

UplinkFast

`Switch(config)#spanning-tree uplinkfast`	Enables UplinkFast.
`Switch(config)#spanning-tree uplinkfast max-update-rate 200`	Enables UplinkFast and sets the update packet rate to 200 packets/second.
	NOTE: UplinkFast cannot be set on an individual VLAN. The **spanning-tree uplinkfast** command affects all VLANs.
	NOTE: For the **max-update-rate** argument, the range is 0 to 32,000 packets/second. The default is 150. If you set the rate to 0, station-learning frames are not generated. This will cause STP to converge more slowly after a loss of connectivity.
`Switch#show spanning-tree summary`	Verifies whether UplinkFast has been enabled.
`Switch#show spanning-tree uplinkfast`	Displays spanning-tree UplinkFast status, which includes maximum update packet rate and participating interfaces.

NOTE: UplinkFast cannot be enabled on VLANs that have been configured for switch priority.

NOTE: UplinkFast is most useful in access layer switches, or switches at the edge of the network. It is not appropriate for backbone devices.

BackboneFast

`Switch(config)#spanning-tree backbonefast`	Enables BackboneFast
`Switch#show spanning-tree summary`	Verifies BackboneFast has been enabled
`Switch#show spanning-tree backbonefast`	Displays spanning-tree BackboneFast status, which includes the number of root link query protocol data units (PDUs) sent/received and number of BackboneFast transitions

Root Guard

You can use Root Guard to limit which switch can become the root bridge. Root Guard should be enabled on all ports where the root bridge is not anticipated, such as access ports.

`Switch(config)#interface fastethernet 0/1`	Moves to interface configuration mode.
`Switch(config-if)#spanning-tree guard root`	Enables Root Guard on the interface.
`Switch#show spanning-tree inconsistentports`	Indicates whether any ports are in a rootinconsistent state.
`Switch(config-if)#show spanning-tree root`	Displays the status and configuration of the root bridge.
	NOTE: The **show spanning-tree root** command output includes root ID for all VLANs, the associated root costs, timer settings, and root ports.
`Switch(config-if)#show spanning-tree`	Displays detailed spanning-tree state and configuration for each VLAN on the switch, including bridge and root IDs, timers, root costs, and forwarding status.

NOTE: You cannot enable both Root Guard and Loop Guard at the same time.

NOTE: Root Guard enabled on an interface applies to all VLANs to which the interface belongs.

NOTE: Do not enable Root Guard on interfaces to be used by the UplinkFast feature.

Loop Guard

Loop Guard is used to prevent alternate or root ports from becoming designated ports due to a failure that leads to a unidirectional link. Loop Guard operates only on interfaces that are considered point-to-point by the spanning tree. Spanning tree determines a port to be point-to-point or shared from the port duplex setting. Loop Guard must be enabled on the nondesignated ports (more precisely, on root and alternate ports) for all possible combinations of active topologies.

NOTE: Both the port duplex and the spanning tree link type can be set manually.

NOTE: You cannot enable both Loop Guard and Root Guard on the same port. The Loop Guard feature is most effective when it is configured on the entire switched network.

`Switch#show spanning-tree active`	Shows which ports are alternate or root ports.
`Switch#show spanning-tree mst`	Shows which ports are alternate or root ports.

`Switch#configure terminal`	Moves to global configuration mode.
`Switch(config)#spanning-tree loopguard default`	Enables Loop Guard globally on the switch for those interfaces that the spanning tree identifies as point to point.
`Switch(config)#interface fastethernet 0/1`	Moves to interface configuration mode.
`Switch(config-if)#spanning-tree guard loop`	Enables loop guard on all the VLANs associated with the selected interface.
`Switch(config)#exit`	Returns to privileged mode.
`Switch#show spanning-tree summary`	Verifies whether Loop Guard has been enabled.
`Switch#show spanning-tree interface detail`	Display spanning-tree link type. A link type of "point to point" is required for Loop Guard.

NOTE: This feature is most effective when it is configured on the entire switched network.

Unidirectional Link Detection

`Switch(config)#udld enable`	Enables unidirectional link detection (UDLD) on all fiber-optic interfaces.
	NOTE: By default, UDLD is disabled.
`Switch(config)#udld aggressive`	Enables UDLD aggressive mode on all fiber interfaces.
`Switch(config)#interface fastethernet 0/24`	Moves to interface configuration mode.
`Switch(config-if)#udld port`	Enables UDLD on this interface (required for copper-based interfaces).
	NOTE: On a fiber-optic (FO) interface, the interface command **udld port** overrides the global command **udld enable**. Therefore, if you issue the command **no udld port** on an FO interface, you will still have the globally enabled **udld enable** command to deal with.
`Switch#show udld`	Displays UDLD information.
`Switch#show udld interface fastethernet 0/1`	Displays UDLD information for interface Fast Ethernet 0/1.
`Switch#udld reset`	Resets all interfaces shut down by UDLD.
	NOTE: You can also use the **shutdown** command, followed by a **no shutdown** command in interface configuration mode, to restart a disabled interface.

Port Error Conditions

A port is "error-disabled" when the switch detects any one of a number of port violations. No traffic is sent or received when the port is in error-disabled state. The **show errdisable detect** command displays a list for the possible err-disable reasons and whether enabled.

The **errdisable detect cause** command allows the network device administrator to enable or disable detection of individual err-disable causes. All causes are enabled by default.

The **errdisable recovery** command enables the network device administrator to configure automatic recovery mechanism variables. This would allow the switch port to again send and receive traffic after a configured period of time if the initial error condition is no longer present. All recovery mechanisms are disabled by default.

`Switch(config)#errdisable recovery cause bpduguard`	Enables the timer for recovery from BPDU Guard error.
`Switch(config)#errdisable recovery interval 3600`	Configures errdisable recovery timer to 3600 seconds.
	NOTE: The same interval is applied to all causes. The range is 30 to 86,400 seconds. The default interval is 300 seconds.
`Switch#show errdisable detect`	Display error-disabled detection status.
`Switch#show errdisable recovery`	Display the error-disabled recovery timer status information.

FlexLinks

`Switch(conf)#interface fastethernet1/0/1`	Moves to interface configuration mode.
`Switch(conf-if)#switchport backup interface fastethernet1/0/2`	Configures FastEthernet 1/0/2 to provide Layer 2 backup to FastEthernet 1/0/1.
`Switch#show interface switchport backup`	Show all the Layer 2 switch backup interface pairs.
	NOTE: FlexLink is an alternative solution to STP.

Changing the Spanning-Tree Mode

You can configure different types of spanning tree on a Cisco switch. The options vary according to the platform:

- **Per-VLAN Spanning Tree (PVST):** There is one instance of spanning tree for each VLAN with Inter-Switch Link (ISL) trunking. This is a Cisco proprietary protocol.

- **Per-VLAN Spanning Tree Plus (PVST+):** There is one instance of spanning tree for each VLAN with 802.1Q trunking. Also Cisco proprietary has added extensions to the PVST protocol.
- **Rapid PVST+:** This mode is the same as PVST+ except that it uses a rapid convergence based on the 802.1w standard.
- **Multiple Spanning Tree (MST):** IEEE 802.1s. Extends the 802.1w Rapid Spanning Tree (RST) algorithm to multiple spanning trees. Multiple VLANs can map to a single instance of RST. You cannot run MST and PVST at the same time.

`Switch(config)#spanning-tree mode mst`	Enables MSTP. This command is available only on a switch running the EI software image.
`Switch(config)#spanning-tree mode pvst`	Enables PVST+. This is the default setting.
`Switch(config)#spanning-tree mode rapid-pvst`	Enables Rapid PVST+.

Extended System ID

`Switch(config)#spanning-tree extend system-id`	Enables extended system ID, also known as MAC address reduction.
	NOTE: Catalyst switches running software earlier than Cisco IOS Release 12.1(8)EA1 do not support the extended system ID.
`Switch#show spanning-tree summary`	Verifies whether extended system ID is enabled.
`Switch#show spanning-tree bridge`	Displays the extended system ID as part of the bridge ID.
	NOTE: The 12-bit extended system ID is the VLAN number for the instance of PVST+ and PVRST+ spanning tree. In MST, these 12 bits carry the instance number.

Enabling Rapid Spanning Tree

`Switch(config)#spanning-tree mode rapid-pvst`	Enables Rapid PVST+.
`Switch(config)#interface fastethernet 0/1`	Moves to interface configuration mode.
`Switch(config-if)#exit`	

`Switch(config)#clear spanning-tree detected-protocols`	
	NOTE: When a current switch running MST or PVRST+ receives a legacy switch 802.1D BPDU, it responds with only IEEE 802.1D BPDUs on that port using a built-in protocol migration mechanism. When the legacy switch is replaced with one running MST or PVRST+, the previous MST/PVRST+ switch still expects to receive 802.1D BPDUs. The **clear spanning-tree detected-protocols** command forces the renegotiation with neighboring switches to restart the protocol migration mechanism.
`Switch#show spanning-tree`	Displays mode, root and bridge IDs, participating ports, and their spanning-tree states.
`Switch#show spanning-tree summary`	Summary of configured port states, including spanning-tree mode.
`Switch#show spanning-tree detail`	Display a detailed summary of spanning-tree interface information, including mode, priority, system ID, MAC address, timers, and role in the spanning tree for each VLAN and port.

Enabling Multiple Spanning Tree

`Switch(config)#spanning-tree mst configuration`	Enters MST configuration mode.
`Switch(config-mst)#instance 1 vlan 4`	Maps VLAN 4 to an Multiple Spanning Tree (MST) instance.
`Switch(config-mst)#instance 1 vlan 1-15`	Maps VLANs 1–15 to MST instance 1.
`Switch(config-mst)#instance 1 vlan 10,20,30`	Maps VLANs 10, 20, and 30 to MST instance 1.
	NOTE: For the **instance** *x* **vlan** *y* command, the instance must be a number between 1 and 15, and the VLAN range is 1 to 4094.

`Switch(config-mst)#name region12`	Specifies the configuration name to be region12.
	NOTE: The **name** argument can be up to 32 characters long and is case sensitive.
`Switch(config-mst)#revision 4`	Specifies the revision number.
	NOTE: The range for the **revision** argument is 0 to 65,535.
`Switch(config-mst)#show current`	Displays the summary of what is currently configured for the MST region.
`Switch(config-mst)#show pending`	Verifies the configuration by displaying a summary of what you have configured for the MST region.
`Switch(config-mst)#exit`	Applies all changes and returns to global configuration mode.
`Switch(config)#spanning-tree mst 1`	Enables spanning-tree mode MST.
	CAUTION: Changing spanning-tree modes can disrupt traffic because all spanning-tree instances are stopped for the old mode and restarted in the new mode.
	NOTE: You cannot run both MSTP and PVST at the same time.
`Switch(config)#spanning-tree mst 1 root primary`	Configures a switch as a primary root switch within MST instance 1. The primary root switch priority is 24,576.
`Switch(config)#spanning-tree mst 1 root secondary`	Configures a switch as a secondary root switch within MST instance 1. The secondary root switch priority is 28,672.
`Switch(config-if)#spanning-tree mst 20 port-priority 0`	Configures a port priority of 0 for MST instance 20.
	NOTE: The priority range is 0 to 240 in increments of 16, where the lower the number, the higher the priority.
`Switch(config-if)#spanning-tree mst 2 cost 250`	Sets the path cost to 250 for MST instance 2 calculations. Path cost is 1 to 200,000,000, with higher values meaning higher costs.
`Switch(config)#exit`	Returns to privileged mode.

Verifying MST

`Switch#show spanning-tree mst configuration`	Displays the MST region configuration
`Switch#show spanning-tree mst configuration digest`	Displays the message digest 5 (MD5) authentication digest included in the current MST configuration identifier (MSTCI)
`Switch#show spanning-tree mst 1`	Displays the MST information for instance 1
`Switch#show spanning-tree mst interface fastethernet 0/1`	Displays the MST information for interface FastEthernet 0/1
`Switch#show spanning-tree mst 1 interface fastethernet 0/1`	Displays the MST information for instance 1 on interface FastEthernet 0/1
`Switch#show spanning-tree mst 1 detail`	Shows detailed information about MST instance 1

Troubleshooting Spanning Tree

`Switch#debug spanning-tree all`	Displays all spanning-tree debugging events
`Switch#debug spanning-tree events`	Displays spanning-tree debugging topology events
`Switch#debug spanning-tree backbonefast`	Displays spanning-tree debugging BackboneFast events
`Switch#debug spanning-tree uplinkfast`	Displays spanning-tree debugging UplinkFast events
`Switch#debug spanning-tree mstp all`	Displays all MST debugging events
`Switch#debug spanning-tree switch state`	Displays spanning-tree port state changes
`Switch#debug spanning-tree pvst+`	Displays PVST+ events

Configuration Example: PVST+

Figure 10-1 shows the network topology for the configuration that follows, which shows how to configure 802.1D-based PVST+ using commands covered in this chapter. All switch-to-switch connections are configured as 802.1Q trunks.

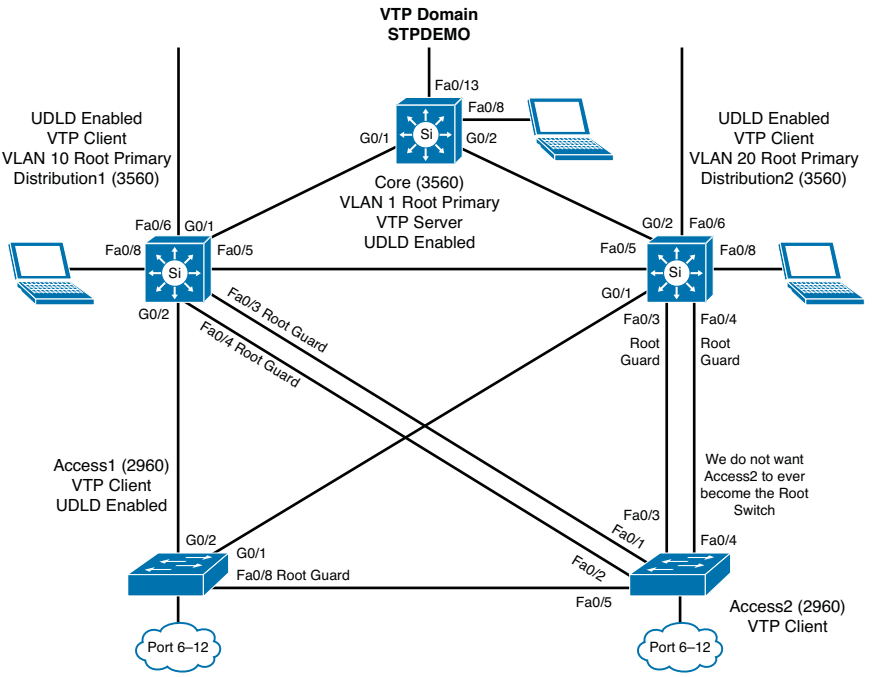


Figure 10-1 Network Topology for STP Configuration Example

Core Switch (3560)

`Switch> enable`	Moves to privileged mode.
`Switch#configure terminal`	Moves to global configuration mode.
`Switch(config)#hostname Core`	Sets hostname.
`Core(config)#no ip domain-lookup`	Turns off Dynamic Name System (DNS) queries so that spelling mistakes will not slow you down.
`Core(config)#vtp mode server`	Changes the switch to VTP server mode. This is the default mode.
`Core(config)#vtp domain STPDEMO`	Configures the VTP domain name to STPDEMO.
`Core(config)#vlan 10`	Creates VLAN 10 and enters VLAN configuration mode.
`Core(config-vlan)#name Accounting`	Assigns a name to the VLAN.
`Core(config-vlan)#exit`	Returns to global configuration mode.
`Core(config)#vlan 20`	Creates VLAN 20 and enters VLAN configuration mode.
`Core(config-vlan)#name Marketing`	Assigns a name to the VLAN.
`Core(config-vlan)#exit`	Returns to global configuration mode.

`Core(config)#spanning-tree vlan 1 root primary`	Configures the switch to become the root switch for VLAN 1.
`Core(config)#udld enable`	Enables UDLD.
`Core(config)#exit`	Returns to privileged mode.
`Core#copy running-config startup-config`	Saves the configuration to NVRAM.

Distribution 1 Switch (3560)

`Switch> enable`	Moves to privileged mode.
`Switch#configure terminal`	Moves to global configuration mode.
`Switch(config)#hostname Distribution1`	Sets hostname.
`Distribution1(config)#no ip domain-lookup`	Turns off DNS queries so that spelling mistakes will not slow you down.
`Distribution1(config)#vtp domain STPDEMO`	Configures the VTP domain name to STPDEMO.
`Distribution1(config)#vtp mode client`	Changes the switch to VTP client mode.
`Distribution1(config)#spanning-tree vlan 10 root primary`	Configures the switch to become the root switch of VLAN 10.
`Distribution1(config)#udld enable`	Enables UDLD on all FO interfaces.
`Distribution1(config)#interface range fastethernet 0/3 - 4`	Moves to interface range mode.
`Distribution1(config-if)#spanning-tree guard root`	Prevents switch on the other end of the link (Access2) from becoming the root switch.
`Distribution1(config-if)#exit`	Returns to global configuration mode.
`Distribution1(config)#exit`	Returns to privileged mode.
`Distribution1#copy running-config startup-config`	Saves the configuration to NVRAM.

Distribution 2 Switch (3560)

`Switch> enable`	Moves to privileged mode
`Switch#configure terminal`	Moves to global configuration mode
`Switch(config)#hostname Distribution2`	Sets hostname
`Distribution2(config)#no ip domain-lookup`	Turns off DNS queries so that spelling mistakes will not slow you down
`Distribution2(config)#vtp domain STPDEMO`	Configures the VTP domain name to STPDEMO
`Distribution2(config)#vtp mode client`	Changes the switch to VTP client mode

`Distribution2(config)#spanning-tree vlan 20 root primary`	Configures the switch to become the root switch of VLAN 20
`Distribution2(config)#udld enable`	Enables UDLD on all FO interfaces
`Distribution2(config)#interface range fastethernet 0/3 - 4`	Moves to interface range mode
`Distribution2(config-if)#spanning-tree guard root`	Prevents the switch on the other end of link (Access2) from becoming the root switch
`Distribution2(config-if)#exit`	Returns to global configuration mode
`Distribution2(config)#exit`	Returns to privileged mode
`Distribution2#copy running-config startup-config`	Saves the configuration to NVRAM

Access 1 Switch (2960)

`Switch> enable`	Moves to privileged mode
`Switch#configure terminal`	Moves to global configuration mode
`Switch(config)#hostname Access1`	Sets hostname
`Access1(config)#no ip domain-lookup`	Turns off DNS queries so that spelling mistakes will not slow you down
`Access1(config)#vtp domain stpdemo`	Configures the VTP domain name to stpdemo
`Access1(config)#vtp mode client`	Changes the switch to VTP client mode
`Access1(config)#interface range fastethernet 0/6 - 12`	Moves to interface range configuration mode
`Access1(config-if-range)#switchport mode access`	Places all interfaces in access mode
`Access1(config-if-range)#spanning-tree portfast`	Places all ports directly into forwarding mode
`Access1(config-if-range)#spanning-tree bpduguard enable`	Enables BPDU Guard
`Access1(config-if-range)#exit`	Moves back to global configuration mode
`Access1(config)#spanning-tree uplinkfast`	Enables UplinkFast to reduce STP convergence time
`Access1(config)#interface fastethernet 0/5`	Moves to interface configuration mode
`Access1(config-if)#spanning-tree guard root`	Prevents the switch on the other end of link (Access2) from becoming the root switch
`Access1(config-if)#exit`	Returns to global configuration mode
`Access1(config)#udld enable`	Enables UDLD on all FO interfaces
`Access1(config)#exit`	Returns to privileged mode
`Access1#copy running-config startup-config`	Saves the configuration to NVRAM

Access 2 Switch (2960)

`Switch> enable`	Moves to privileged mode
`Switch#configure terminal`	Moves to global configuration mode
`Switch(config)#hostname Access2`	Sets hostname
`Access2(config)#no ip domain-lookup`	Turns off DNS queries so that spelling mistakes will not slow you down
`Access2(config)#vtp domain stpdemo`	Configures the VTP domain name to stpdemo
`Access2(config)#vtp mode client`	Changes the switch to VTP client mode
`Access2(config)#interface range fastethernet 0/6 - 12`	Moves to interface range configuration mode
`Access2(config-if-range)#switchport mode access`	Places all interfaces in access mode
`Access2(config-if-range)#spanning-tree portfast`	Places all ports directly into forwarding mode
`Access2(config-if-range)#spanning-tree bpduguard enable`	Enables BPDU Guard
`Access2(config-if-range)#exit`	Moves back to global configuration mode
`Access2(config)#spanning-tree vlan 1,10,20 priority 61440`	Ensures this switch will not become the root switch for VLAN 10
`Access2(config)#exit`	Returns to privileged mode
`Access2#copy running-config startup-config`	Saves the configuration to NVRAM

Spanning-Tree Migration Example: PVST+ to Rapid-PVST+

The topology in Figure 10-1 is used for this migration example and adds to the configuration of the previous example.

Rapid PVST+ uses the same BPDU format as the 802.1D. This interoperability between the two spanning-tree protocols enables a longer conversion time in large networks without disrupting services.

The spanning-tree features UplinkFast and BackboneFast in 802.1D-based PVST+ are already incorporated in the 802.1w-based Rapid PVST+ and are disabled when you enable Rapid PVST+. The 802.1D-based features of PVST+ such as PortFast, BPDU Guard, BPDU Filter, Root Guard, and Loop Guard are applicable in Rapid PVST+ mode and need not be changed.

Access 1 Switch (2960)

`Access1>` **`enable`**	Moves to privileged mode
`Access1#`**`configure terminal`**	Moves to global configuration mode
`Access1 (config)#`**`spanning-tree mode rapid-pvst`**	Enables 802.1w-based Rapid PVST+
`Access1(config)#`**`no spanning-tree uplinkfast`**	Removes UplinkFast programming line
`Access1(config)#`**`no spanning-tree backbonefast`**	Removes BackboneFast programming line

Access 2 Switch (2960)

`Access2>` **`enable`**	Moves to privileged mode
`Access2#`**`configure terminal`**	Moves to global configuration mode
`Access2 (config)#`**`spanning-tree mode rapid-pvst`**	Enables 802.1w-based Rapid PVST+

Distribution 1 Switch (3560)

`Distribution1>` **`enable`**	Moves to privileged mode
`Distribution1#`**`configure terminal`**	Moves to global configuration mode
`Distribution1 (config)#`**`spanning-tree mode rapid-pvst`**	Enables 802.1w-based Rapid PVST+

Distribution 2 Switch (3560)

`Distribution2>` **`enable`**	Moves to privileged mode
`Distribution2#`**`configure terminal`**	Moves to global configuration mode
`Distribution2 (config)#`**`spanning-tree mode rapid-pvst`**	Enables 802.1w-based Rapid PVST+

Core Switch (3560)

`Core>` **`enable`**	Moves to privileged mode
`Core#`**`configure terminal`**	Moves to global configuration mode
`Core(config)#`**`spanning-tree mode rapid-pvst`**	Enables 802.1w-based Rapid PVST+

CHAPTER 11

Implementing Inter-VLAN Routing

This chapter provides information about the following topics:

- Inter-VLAN communication using an external router: router-on-a-stick
- Inter-VLAN routing tips
- Removing L2 switchport capability of a switch port
- Configuring SVI autostate
- Inter-VLAN communication on a multilayer switch through a switch virtual interface
- Configuration example: Inter-VLAN communication
- Configuration example: IPv6 Inter-VLAN communication

Inter-VLAN Communication Using an External Router: Router-on-a-Stick

`Router(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#duplex full`	Sets interface to full duplex.
`Router(config-if)#no shutdown`	Enables interface.
`Router(config-if)#interface fastethernet0/0.1`	Creates subinterface 0/0.1 and moves to subinterface configuration mode.
`Router(config-subif)#description Management VLAN 1`	(Optional) Sets locally significant descriptor of the subinterface.
`Router(config-subif)#encapsulation dot1q 1 native`	Assigns VLAN 1 to this subinterface. VLAN 1 will be the native VLAN. This subinterface will use the 802.1Q trunking protocol.
`Router(config-subif)#ip address 192.168.1.1 255.255.255.0`	Assigns IP address and netmask.
`Router(config-subif)#interface fastethernet0/0.10`	Creates subinterface 0/0.10 and moves to subinterface configuration mode.
`Router(config-subif)#description Accounting VLAN 10`	(Optional) Sets locally significant descriptor of the subinterface.
`Router(config-subif)#encapsulation dot1q 10`	Assigns VLAN 10 to this subinterface. This subinterface will use the 802.1Q trunking protocol.
`Router(config-subif)#ip address 192.168.10.1 255.255.255.0`	Assigns IP address and netmask.

`Router(config-subif)#exit`	Returns to interface configuration mode.
`Router(config-if)#exit`	Returns to global configuration mode.
`Router(config)#`	

NOTE: The subnets of the VLANs are directly connected to the router. Routing between these subnets does not require a dynamic routing protocol. In a more complex topology, these routes would need to either be advertised with whatever dynamic routing protocol is being used, or be redistributed into whatever dynamic routing protocol is being used.

NOTE: Routes to the subnets associated with these VLANs will appear in the routing table as directly connected networks.

Inter-VLAN Routing Tips

- Although most routers support both Inter-Switch Link (ISL) and Dot1Q encapsulation, some switch models support only Dot1Q, such as the 2950 and 2960 series.
- If you need to use ISL as your trunking protocol, use the command **encapsulation isl** *x*, where *x* is the number of the VLAN to be assigned to that subinterface.
- Recommended best practice is to use the same number of the VLAN number for the subinterface number. It is easier to troubleshoot VLAN 10 on subinterface fa0/0.10 than on fa0/0.2
- The native VLAN (usually VLAN 1) cannot be configured on a subinterface for Cisco IOS releases that are earlier than 12.1(3)T. Native VLAN IP addresses will therefore need to be configured on the physical interface. Other VLAN traffic will be configured on subinterfaces:

```
Router(config)#interface fastethernet0/0
Router(config-if)#encapsulation dot1q 1 native
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#interface fastethernet0/0.10
Router(config-subif)#encapsulation dot1q 10
Router(config-subif)#ip address 192.168.10.1 255.255.255.0
```

Removing L2 Switch Port Capability of a Switch Port

`3750Switch(config)#interface fastethernet0/1`	Moves to interface configuration mode.
`3750Switch(config-if)#no switchport`	Creates a Layer 3 port on the switch. Also known as a *routed switch port*.
	NOTE: The **no switchport** command can be used on physical ports only on a Layer 3-capable switch.

Configuring SVI Autostate

`3750Switch(config)#interface fastethernet0/1`	Moves to interface configuration mode.
`3750Switch(config-if)#switchport auto-state exclude`	Excludes the access port/trunk in defining the status of an SVI as line up or down.
	NOTE: This command is commonly used for ports that are used for monitoring (for instance, so that a monitoring port did not cause the SVI to remain up when no other ports are active in the VLAN).

NOTE: For the SVI line state to be up, at least one port in the VLAN must be up and forwarding. The **switchport autostate exclude** command excludes a port from the SVI interface line-state up or down calculation.

Inter-VLAN Communication on a Multilayer Switch Through a Switch Virtual Interface

Rather than using an external router to provide inter-VLAN communication, a multilayer switch can perform the same task through the use of a switched virtual interface (SVI).

`3750Switch(config)#ip routing`	Enables routing on the switch
`3750Switch(config)#interface vlan 1`	Creates a virtual interface for VLAN 1 and enters interface configuration mode
`3750Switch(config-if)#ip address 172.16.1.1 255.255.255.0`	Assigns IP address and netmask
`3750Switch(config-if)#no shutdown`	Enables the interface
`3750Switch(config)#interface vlan 10`	Creates a virtual interface for VLAN 10 and enters interface configuration mode
`3750Switch(config-if)#ip address 172.16.10.1 255.255.255.0`	Assigns IP address and netmask
`3750Switch(config-if)#no shutdown`	Enables the interface
`3750Switch(config)#interface vlan 20`	Creates a virtual interface for VLAN 20 and enters interface configuration mode
`3750Switch(config-if)#ip address 172.16.20.1 255.255.255.0`	Assigns IP address and netmask
`3750Switch(config-if)#no shutdown`	Enables the interface
`3750Switch(config-if)#exit`	Returns to global configuration mode

Configuration Example: Inter-VLAN Communication

Figure 11-1 shows the network topology for the configuration that follows, which shows how to configure inter-VLAN communication using commands covered in this chapter. Some commands used in this configuration are from previous chapters.

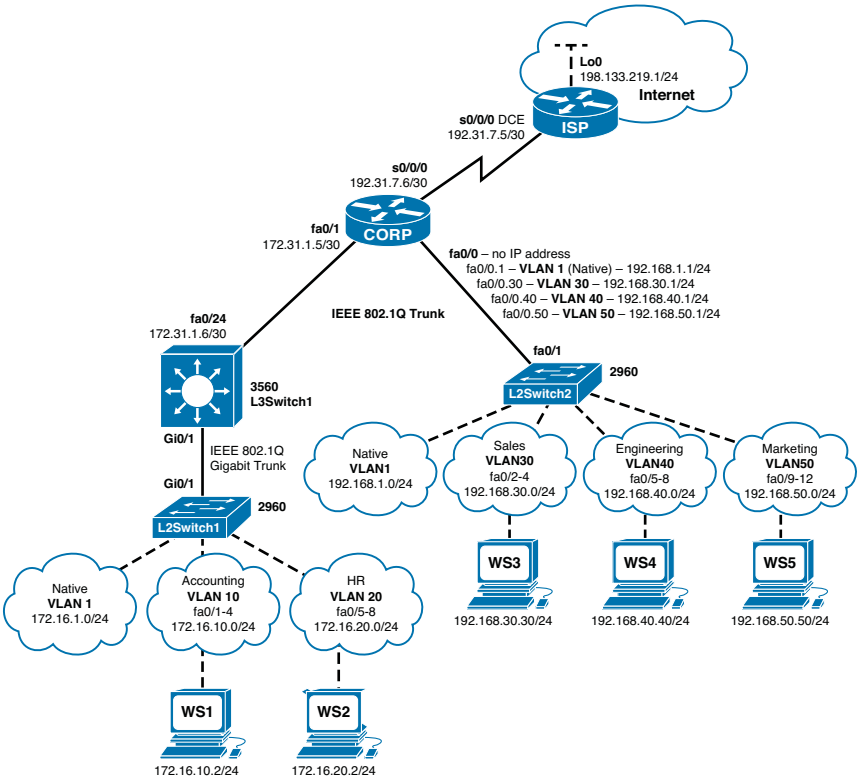


Figure 11-1 Network Topology for Inter-VLAN Communication Configuration

ISP Router

`Router> enable`	Moves to privileged mode.
`Router>#configure terminal`	Moves to global config mode.
`Router(config)#hostname ISP`	Sets hostname.
`ISP(config)#interface loopback0`	Moves to interface configuration mode.
`ISP(config-if)#description simulated address representing remote website`	Sets locally significant interface description.
`ISP(config-if)#ip address 198.133.219.1 255.255.255.0`	Assigns IP address and netmask.

ISP(config-if)#**interface serial0/0/0**	Moves to interface configuration mode.
ISP(config-if)#**description WAN link to the Corporate Router**	Sets locally significant interface description.
ISP(config-if)#**ip address 192.31.7.5 255.255.255.252**	Assigns IP address and netmask.
ISP(config-if)#**clock rate 56000**	Assigns a clock rate to the interface. (The DCE cable is plugged into this interface.)
ISP(config-if)#**no shutdown**	Enables the interface.
ISP(config-if)#**exit**	Returns to global configuration mode.
ISP(config-if)#**router eigrp 10**	Creates Enhanced Interior Gateway Routing Protocol (EIGRP) routing process 10.
ISP(config-router)#**network 198.133.219.0**	Advertises directly connected networks (classful address only).
ISP(config-router)#**network 192.31.7.0**	Advertises directly connected networks (classful address only).
ISP(config-router)#**no auto-summary**	Disables autosummarization.
ISP(config-router)#**exit**	Returns to global configuration mode.
ISP(config)#**exit**	Returns to privileged mode.
ISP#**copy running-config startup-config**	Saves the configuration to NVRAM.

CORP Router

Router> **enable**	Moves to privileged mode.
Router>#**configure terminal**	Moves to global configuration mode.
Router(config)#**hostname CORP**	Sets hostname.
CORP(config)#**no ip domain-lookup**	Turns off Domain Name System (DNS) resolution to avoid wait time due to DNS lookup of spelling errors.
CORP(config)#**interface serial0/0/0**	Moves to interface configuration mode.
CORP(config-if)#**description link to ISP**	Sets locally significant interface description.
CORP(config-if)#**ip address 192.31.7.6 255.255.255.252**	Assigns IP address and netmask.
CORP(config-if)#**no shutdown**	Enables interface.
CORP(config)#**interface fastethernet0/1**	Moves to interface configuration mode.
CORP(config-if)#**description link to 3560 Switch**	Sets locally significant interface description.

`CORP(config-if)#ip address 172.31.1.5 255.255.255.252`	Assigns IP address and netmask.
`CORP(config-if)#no shutdown`	Enables interface.
`CORP(config-if)#exit`	Returns to global configuration mode.
`CORP(config)#interface fastethernet0/0`	Enters interface configuration mode.
`CORP(config-if)#duplex full`	Enables full-duplex operation to ensure trunking will take effect between here and L2Switch2.
`CORP(config-if)#no shutdown`	Enables interface.
`CORP(config-if)#interface fastethernet0/0.1`	Creates a virtual subinterface and moves to subinterface configuration mode.
`CORP(config-subif)#description Management VLAN 1 - Native VLAN`	Sets locally significant interface description.
`CORP(config-subif)#encapsulation dot1q 1 native`	Assigns VLAN 1 to this subinterface. VLAN 1 will be the native VLAN. This subinterface will use the 802.1Q trunking protocol.
`CORP(config-subif)#ip address 192.168.1.1 255.255.255.0`	Assigns IP address and netmask.
`CORP(config-subif)#interface fastethernet 0/0.30`	Creates a virtual subinterface and moves to subinterface configuration mode.
`CORP(config-subif)#description Sales VLAN 30`	Sets locally significant interface description.
`CORP(config-subif)#encapsulation dot1q 30`	Assigns VLAN 30 to this subinterface. This subinterface will use the 802.1Q trunking protocol.
`CORP(config-subif)#ip address 192.168.30.1 255.255.255.0`	Assigns IP address and netmask.
`CORP(config-subif)#interface fastethernet 0/0.40`	Creates a virtual subinterface and moves to subinterface configuration mode.
`CORP(config-subif)#description Engineering VLAN 40`	Sets locally significant interface description.
`CORP(config-subif)#encapsulation dot1q 40`	Assigns VLAN 40 to this subinterface. This subinterface will use the 802.1Q trunking protocol.
`CORP(config-subif)#ip address 192.168.40.1 255.255.255.0`	Assigns IP address and netmask.
`CORP(config-subif)#interface fastethernet 0/0.50`	Creates a virtual subinterface and moves to subinterface configuration mode.
`CORP(config-subif)#description Marketing VLAN 50`	Sets locally significant interface description.

CORP(config-subif)#**encapsulation dot1q 50**	Assigns VLAN 50 to this subinterface. This subinterface will use the 802.1Q trunking protocol.
CORP(config-subif)#**ip add 192.168.50.1 255.255.255.0**	Assigns IP address and netmask.
CORP(config-subif)#**exit**	Returns to interface configuration mode.
CORP(config-if)#**exit**	Returns to global configuration mode.
CORP(config)#**router eigrp 10**	Creates EIGRP routing process 10 and moves to router configuration mode.
CORP(config-router)#**network 192.168.1.0**	Advertises the 192.168.1.0 network.
CORP(config-router)#**network 192.168.30.0**	Advertises the 192.168.30.0 network.
CORP(config-router)#**network 192.168.40.0**	Advertises the 192.168.40.0 network.
CORP(config-router)#**network 192.168.50.0**	Advertises the 192.168.50.0 network.
CORP(config-router)#**network 172.31.0.0**	Advertises the 172.31.0.0 network.
CORP(config-router)#**network 192.31.7.0**	Advertises the 192.31.7.0 network.
CORP(config-router)#**no auto-summary**	Turns off automatic summarization at classful boundary.
CORP(config-router)#**exit**	Returns to global configuration mode.
CORP(config)#**exit**	Returns to privileged mode.
CORP#**copy running-config startup-config**	Saves the configuration in NVRAM.

L2Switch2 (Catalyst 2960)

Switch>**enable**	Moves to privileged mode.
Switch#**configure terminal**	Moves to global configuration mode.
Switch(config)#**hostname L2Switch2**	Sets hostname.
L2Switch2(config)#**no ip domain-lookup**	Turns off DNS resolution.
L2Switch2(config)#**vlan 30**	Creates VLAN 30 and enters VLAN configuration mode.
L2Switch2(config-vlan)#**name Sales**	Assigns a name to the VLAN.
L2Switch2(config-vlan)#**exit**	Returns to global configuration mode.
L2Switch2(config)#**vlan 40**	Creates VLAN 40 and enters VLAN configuration mode.
L2Switch2(config-vlan)#**name Engineering**	Assigns a name to the VLAN.

`L2Switch2(config-vlan)#vlan 50`	Creates VLAN 50 and enters VLAN configuration mode. Note that you do not have to exit back to global configuration mode to execute this command.
`L2Switch2(config-vlan)#name Marketing`	Assigns a name to the VLAN.
`L2Switch2(config-vlan)#exit`	Returns to global configuration mode.
`L2Switch2(config)#interface range fastethernet 0/2 - 4`	Enables you to set the same configuration parameters on multiple ports at the same time.
`L2Switch2(config-if-range)#switchport mode access`	Sets ports 2–4 as access ports.
`L2Switch2(config-if-range)#switchport access vlan 30`	Assigns ports 2–4 to VLAN 30.
`L2Switch2(config-if-range)#interface range fastethernet 0/5 - 8`	Enables you to set the same configuration parameters on multiple ports at the same time.
`L2Switch2(config-if-range)#switchport mode access`	Sets ports 5–8 as access ports.
`L2Switch2(config-if-range)#switchport access vlan 40`	Assigns ports 5–8 to VLAN 40.
`L2Switch2(config-if-range)#interface range fastethernet 0/9 - 12`	Enables you to set the same configuration parameters on multiple ports at the same time.
`L2Switch2(config-if-range)#switchport mode access`	Sets ports 9–12 as access ports.
`L2Switch2(config-if-range)#switchport access vlan 50`	Assigns ports 9–12 to VLAN 50.
`L2Switch2(config-if-range)#exit`	Returns to global configuration mode.
`L2Switch2(config)#int fastethernet 0/1`	Moves to interface configuration mode.
`L2Switch2(config)#description Trunk Link to CORP Router`	Sets locally significant interface description.
`L2Switch2(config-if)#switchport mode trunk`	Puts the interface into trunking mode and negotiates to convert the link into a trunk link.
`L2Switch2(config-if)#exit`	Returns to global configuration mode.
`L2Switch2(config)#interface vlan 1`	Creates virtual interface for VLAN 1 and enters interface configuration mode.
`L2Switch2(config-if)#ip address 192.168.1.2 255.255.255.0`	Assigns IP address and netmask.
`L2Switch2(config-if)#no shutdown`	Enables interface.
`L2Switch2(config-if)#exit`	Returns to global configuration mode.

L2Switch2(config)#**ip default-gateway 192.168.1.1**	Assigns default gateway address.
L2Switch2(config)#**exit**	Returns to privileged mode.
L2Switch2#**copy running-config startup-config**	Saves the configuration in NVRAM.

L3Switch1 (Catalyst 3560)

Switch>**enable**	Moves to privileged mode
Switch#**configure terminal**	Moves to global configuration mode
Switch(config)#**hostname L3Switch1**	Sets hostname
L3Switch1(config)#**no ip domain-lookup**	Turns off DNS queries so that spelling mistakes will not slow you down
L3Switch1(config)#**vtp mode sever**	Changes the switch to VTP server mode
L3Switch1(config)#**vtp domain testdomain**	Configures the VTP domain name to testdomain
L3Switch1(config)#**vlan 10**	Creates VLAN 10 and enters VLAN configuration mode
L3Switch1(config-vlan)#**name Accounting**	Assigns a name to the VLAN
L3Switch1(config-vlan)#**exit**	Returns to global configuration mode
L3Switch1(config)#**vlan 20**	Creates VLAN 20 and enters VLAN configuration mode
L3Switch1(config-vlan)#**name HR**	Assigns a name to the VLAN
L3Switch1(config-vlan)#**exit**	Returns to global configuration mode
L3Switch1(config)#**interface gigabitethernet0/1**	Moves to interface configuration mode
L3Switch1(config-if)#**switchport trunk encapsulation dot1q**	Specifies 802.1Q encapsulation on the trunk link
L3Switch1(config-if)#**switchport mode trunk**	Puts the interface into trunking mode and negotiates to convert the link into a trunk link
L3Switch1(config-if)#**exit**	Returns to global configuration mode
L3Switch1(config)#**ip routing**	Enables IP routing on this device
L3Switch1(config)#**interface vlan 1**	Creates virtual interface for VLAN 1 and enters interface configuration mode
L3Switch1(config-if)#**ip address 172.16.1.1 255.255.255.0**	Assigns IP address and netmask
L3Switch1(config-if)#**no shutdown**	Enables interface
L3Switch1(config-if)#**interface vlan 10**	Creates virtual interface for VLAN 10 and enters interface configuration mode

`L3Switch1(config-if)#ip address 172.16.10.1 255.255.255.0`	Assigns IP address and mask
`L3Switch1(config-if)#no shutdown`	Enables interface
`L3Switch1(config-if)#interface vlan 20`	Creates virtual interface for VLAN 20 and enters interface configuration mode
`L3Switch1(config-if)#ip address 172.16.20.1 255.255.255.0`	Assigns IP address and mask
`L3Switch1(config-if)#no shutdown`	Enables interface
`L3Switch1(config-if)#exit`	Returns to global configuration mode
`L3Switch1(config)#interface fastethernet 0/24`	Enters interface configuration mode
`L3Switch1(config-if)#no switchport`	Creates a Layer 3 port on the switch
`L3Switch1(config-if)#ip address 172.31.1.6 255.255.255.252`	Assigns IP address and netmask
`L3Switch1(config-if)#exit`	Returns to global configuration mode
`L3Switch1(config)#router eigrp 10`	Creates EIGRP routing process 10 and moves to router config mode
`L3Switch1(config-router)#network 172.16.0.0`	Advertises the 172.16.0.0 classful network
`L3Switch1(config-router)#network 172.31.0.0`	Advertises the 172.31.0.0 classful network
`L3Switch1(config-router)#no auto-summary`	Turns off automatic summarization at classful boundary
`L3Switch1(config-router)#exit`	Applies changes and returns to global configuration mode
`L3Switch1(config)#exit`	Returns to privileged mode
`L3Switch1#copy running-config startup-config`	Saves configuration in NVRAM

L2Switch1 (Catalyst 2960)

`Switch>enable`	Moves to privileged mode
`Switch#configure terminal`	Moves to global configuration mode
`Switch(config)#hostname L2Switch1`	Sets hostname
`L2Switch1(config)#no ip domain-lookup`	Turns off DNS queries so that spelling mistakes will not slow you down
`L2Switch1(config)#vtp domain testdomain`	Configures the VTP domain name to testdomain
`L2Switch1(config)#vtp mode client`	Changes the switch to VTP client mode
`L2Switch1(config)#interface range fastethernet 0/1 - 4`	Enables you to set the same configuration parameters on multiple ports at the same time

L2Switch1(config-if-range)#**switchport mode access**	Sets ports 1–4 as access ports
L2Switch1(config-if-range)#**switchport access vlan 10**	Assigns ports 1–4 to VLAN 10
L2Switch1(config-if-range)#**interface range fastethernet 0/5 - 8**	Enables you to set the same configuration parameters on multiple ports at the same time
L2Switch1(config-if-range)#**switchport mode access**	Sets ports 5–8 as access ports
L2Switch1(config-if-range)#**switchport access vlan 20**	Assigns ports 5–8 to VLAN 20
L2Switch1(config-if-range)#**exit**	Returns to global configuration mode
L2Switch1(config)#**interface gigabitethernet0/1**	Moves to interface configuration mode
L2Switch1(config-if)#**switchport mode trunk**	Puts the interface into trunking mode and negotiates to convert the link into a trunk link
L2Switch1(config-if)#**exit**	Returns to global configuration mode
L2Switch1(config)#**interface vlan 1**	Creates virtual interface for VLAN 1 and enters interface configuration mode
L2Switch1(config-if)#**ip address 172.16.1.2 255.255.255.0**	Assigns IP address and netmask
L2Switch1(config-if)#**no shutdown**	Enables interface
L2Switch1(config-if)#**exit**	Returns to global configuration mode
L2Switch1(config)#**ip default-gateway 172.16.1.1**	Assigns default gateway address
L2Switch1(config)#**exit**	Returns to privileged mode
L2Switch1#**copy running-config startup-config**	Saves the configuration in NVRAM

Configuration Example: IPv6 Inter-VLAN Communication

Figure 11-2 shows the network topology for the configuration that follows, which shows how to configure IPv6 inter-VLAN communication using commands covered in this chapter. Some commands used in this configuration are from previous chapters.

CORP and ISP routers are Cisco CISCO2911/K9 running c2900-universalk9-mz.SPA.152-4.M2.bin with ipbasek9, securityk9, and uck9 feature sets enabled.

L3Sw1 is a Cisco WS-C3560V2-24PS running c3560-ipservicesk9-mz.150-2.SE6.bin.

L2Sw1 and L2Sw2 are Cisco WS-C2960+24TC-L switches running c2960-lanbasek9-mz.150-2.EZ.bin.

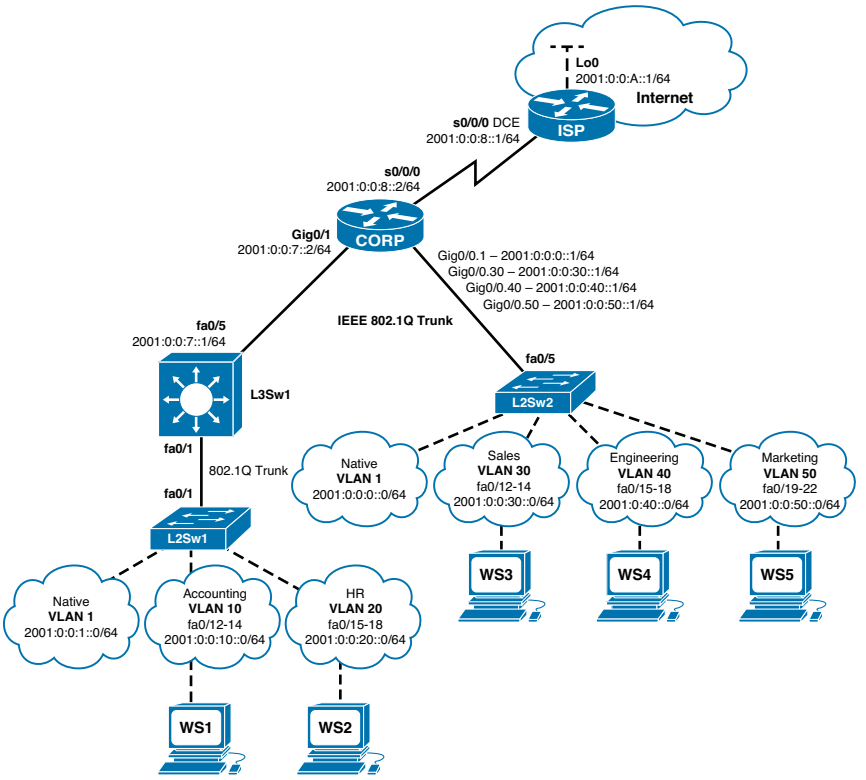


Figure 11-2 Network Topology for IPv6 Inter-VLAN Communication Configuration

ISP Router

`ISP(config)#hostname ISP`	Configures the router name.
`ISP(config)#ipv6 unicast-routing`	Enables IPv6 routing
`ISP(config)#interface loopback0`	Enters interface configuration mode.
`ISP(config-if)#ipv6 address 2001:0:0:A::1/64`	Assigns an IPv6 address
`ISP(config-if)#interface serial0/0/0`	Enters interface configuration mode.
`ISP(config-if)#ipv6 address 2001:0:0:8::1/64`	Assigns IPv6 address.
`ISP(config-if)#no shutdown`	Turns on this interface.
`ISP(config-if)#exit`	Exits into global configuration mode.
`ISP(config)#ipv6 route ::/0 serial0/0/0`	Creates a default static route to return traffic from the Internet.
	NOTE: A dynamic routing protocol can also be used here as well.
`ISP(config)#end`	Returns to privileged EXEC mode.

CORP Router

`Router(config)#hostname CORP`	Assigns name to the router
`CORP(config)#ipv6 unicast-routing`	Enables global IPv6 forwarding
`CORP(config)#ipv6 router ospf 1`	Enters OSPFv3 programming mode
`CORP(config-rtr)#router-id 192.168.1.1`	Assigns a router ID for the OSPFv3 process
`CORP(config-rtr)#default-information originate`	Adds any default routing information to the OSPFv3 updates
`CORP(config-rtr)#exit`	Exits to global configuration mode
`CORP(config)#interface gigabitethernet0/0.1`	Enters subinterface programming mode
`CORP(config-subif)#encapsulation dot1Q 1 native`	Assigns 8021q as the trunking protocol and associates VLAN 1 to this subinterface
`CORP(config-subif)#ipv6 address 2001::1/64`	Assigns an IPv6 address
`CORP(config-subif)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3
`CORP(config-subif)#interface gigabitethernet0/0.30`	Enters subinterface programming mode
`CORP(config-subif)#encapsulation dot1Q 30`	Assigns 8021q as the trunking protocol and associate VLAN 30 to this subinterface
`CORP(config-subif)#ipv6 address 2001:0:0:30::1/64`	Assigns an IPv6 address
`CORP(config-subif)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3
`CORP(config-subif)#interface gigabitethernet0/0.40`	Enters subinterface programming mode
`CORP(config-subif)#encapsulation dot1Q 40`	Assigns 8021q as the trunking protocol and associate VLAN 40 to this subinterface
`CORP(config-subif)#ipv6 address 2001:0:0:40::1/64`	Assigns an IPv6 address
`CORP(config-subif)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3
`CORP(config-subif)#interface gigabitethernet0/0.50`	Enters subinterface programming mode
`CORP(config-subif)#encapsulation dot1Q 50`	Assigns 8021q as the trunking protocol and associate VLAN 50 to this subinterface
`CORP(config-subif)#ipv6 address 2001:0:0:50::1/64`	Assigns an IPv6 address

`CORP(config-subif)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3
`CORP(config-subif)#interface gigabitethernet0/1`	Enters interface programming mode
`CORP(config-if)#ipv6 address 2001:0:0:7::2/64`	Assigns an IPv6 address
`CORP(config-if)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3
`CORP(config-if)#interface gigabitethernet0/0`	Enters interface programming mode
`CORP(config-if)#no shutdown`	Turn this interface on
`CORP(config-if)#interface serial0/0/0`	Enters interface programming mode
`CORP(config-if)#ipv6 address 2001:0:0:8::2/64`	Assigns an IPv6 address
`CORP(config-if)#clock rate 8000000`	Specifies a clock rate for this serial DCE interface
`CORP(config-if)#no shutdown`	Turn this interface on
`CORP(config-if)#exit`	Exits to global configuration programming mode
`CORP(config)#ipv6 route ::/0 Serial0/0/0`	Creates a default static route pointing to the ISP
`CORP(config)#end`	Returns to privileged EXEC mode

L2Sw2 (Catalyst 2960)

`Switch(config)#hostname L2Sw2`	Assigns the switch device name.
`L2Sw2(config)#sdm prefer dual-ipv4-and-ipv6 default`	Configures the Switching Database Manager 9SDM) on the switch to optimize memory and operating system for both IPv4 and IPv6 Layer 3 forwarding.
	NOTE: If this is a change in the SDM settings, the switch must be reloaded for this change to take effect.
`L2Sw2(config)#vlan 30,40,50`	Creates VLANs 30, 40, and 50.
`L2Sw2(config-vlan)# exit`	Exits VLAN configuration mode.
`L2Sw2(config)#interface fastethernet0/5`	Enters switchport interface configuration mode.
`L2Sw2(config-if)#switchport mode trunk`	Sets this port to trunk unconditionally.
`L2Sw2(config-if)#interface range fastethernet0/12 - 14`	Enters switchport configuration mode for a range of switch ports.

`L2Sw2(config-if-range)#switchport mode access`	Sets these ports to be access ports.
`L2Sw2(config-if-range)# switchport access vlan 30`	Assigns these ports to VLAN 30.
`L2Sw2(config-if-range)#interface range fastethernet0/15 - 18`	Enters switchport configuration mode for a range of switch ports.
`L2Sw2(config-if-range)#switchport mode access`	Sets these ports to be access ports.
`L2Sw2(config-if-range)# switchport access vlan 40`	Assigns these ports to VLAN 20.
`L2Sw2(config-if-range)#interface range fastethernet0/19 - 22`	Enters switchport configuration mode for a range of switchports.
`L2Sw2(config-if-range)#switchport mode access`	Sets these ports to be access ports.
`L2Sw2(config-if-range)# switchport access vlan 50`	Assigns these ports to VLAN 50.
`L2Sw2(config-if-range)#interface vlan1`	Enters interface configuration mode for the management VLAN.
`L2Sw2(config-if)# ipv6 address 2001::2/64`	Assigns an IPv6 address.
`L2Sw2(config-if)#no shutdown`	Turn the interface on.
`L2Sw2(config-if)#exit`	Exits to global configuration mode.
`L2Sw2(config)#end`	Returns to privileged EXEC mode.

L3Sw1 (Catalyst 3560)

`Switch(config)#hostname L3Sw1`	Assigns the switch name.
`L3Sw1(config)#sdm prefer dual-ipv4-and-ipv6 routing`	Configures the Switching Database Manager on the switch to optimize memory and operating system for both IPv4 and IPv6 Layer 3 forwarding.
`L3Sw1(config)#ipv6 unicast-routing`	Enables IPv6 forwarding.
`L3Sw1(config)#vlan 10,20`	Creates VLANs 10 and 20.
`L3Sw1(config-vlan)#exit`	Exits VLAN configuration mode.
`L3Sw1(config)#interface fastethernet0/1`	Enters interface configuration mode.
`L3Sw1(config-if)#switchport trunk encapsulation dot1q`	Define 802.1Q as the trunking protocol.
`L3Sw1(config-if)#switchport mode trunk`	Sets this port to trunk unconditionally.
`L3Sw1(config-if)#ipv6 router ospf 1`	Enters OSPFv3 configuration mode.

`L3Sw1(config-rtr)#router-id 192.168.1.2`	Assigns the OSPFv3 router ID.
`L3Sw1(config-rtr)#exit`	Exits to global configuration mode.
`L3Sw1(config)#interface fastethernet0/5`	Enters switchport interface configuration mode.
`L3Sw1(config-if)#no switchport`	Changes this Layer 2 switch port to a Layer 3 routed port.
`L3Sw1(config-if)#ipv6 address 2001:0:0:7::1/64`	Assigns an IPv6 address.
`L3Sw1(config-if)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3.
`L3Sw1(config-if)#interface vlan1`	Enters interface configuration mode for VLAN 1.
`L3Sw1(config-if)#ipv6 address 2001:0:0:1::1/64`	Assigns an IPv6 address.
`L3Sw1(config-if)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3.
`L3Sw1(config-if)#interface vlan10`	Enters interface configuration mode for VLAN 10.
`L3Sw1(config-if)#ipv6 address 2001:0:0:10::1/64`	Assigns an IPv6 address.
`L3Sw1(config-if)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3.
`L3Sw1(config-if)#interface vlan20`	Enters interface configuration mode for VLAN 10.
`L3Sw1(config-if)#ipv6 address 2001:0:0:20::1/64`	Assigns an IPv6 address.
`L3Sw1(config-if)#ipv6 ospf 1 area 0`	Specifies this as an interface that will participate in OSPFv3.
`L3Sw1(config-if)#end`	Returns to privileged EXEC mode.

L2Sw1 (Catalyst 2960)

`Switch(config)#hostname L2Sw1`	Assigns device name for L2Sw1
`L2Sw1(config)#sdm prefer dual-ipv4-and-ipv6 default`	Configures the Switching Database Manager on the switch to optimize memory and operating system for both IPv4 and IPv6 Layer 3 forwarding
`L2Sw1(config)#vlan 10,20`	Creates VLAN 10 and 20
`L2Sw1(config-vlan)#exit`	Exits VLAN configuration mode
`L2Sw1(config)#interface fastethernet0/1`	Enters switchport interface configuration mode

`L2Sw1(config-if)#switchport mode trunk`	Sets this port to trunk unconditionally
`L2Sw1(config-if)#interface range fastethernet0/12 - 14`	Enters switchport configuration mode for a range of switch ports
`L2Sw1(config-if-range)#switchport mode access`	Sets these ports to be access ports
`L2Sw1(config-if-range)#switchport access vlan 10`	Assigns these ports to VLAN 10
`L2Sw1(config-if-range)#interface range fastethernet0/15 - 18`	Enters switchport configuration mode for a range of switch ports
`L2Sw1(config-if-range)#switchport mode access`	Sets these ports to be access ports
`L2Sw1(config-if-range)#switchport access vlan 20`	Assigns these ports to VLAN 20
`L2Sw1(config-if-range)#interface vlan1`	Moves to interface configuration mode
`L2Sw1(config-if)#ipv6 address 2001:0:0:4::2/64`	Assigns an IPv6 address
`L2Sw1(config-if)#exit`	Returns to global configuration mode
`L2Sw1(config)#end`	Returns to privileged EXEC mode

This page intentionally left blank

CHAPTER 12

Implementing High-Availability Networks

This chapter provides information about the following topics:

- Configuring IP service level agreements
  - Configuring authentication for IP SLA
  - Monitoring IP SLA operations
- Implementing port mirroring
  - Default SPAN and RSPAN configuration
  - Local SPAN guidelines for configuration
  - Configuring local SPAN
  - Remote SPAN guidelines for configuration
  - Configuring remote SPAN
  - Verifying and troubleshooting local and remote SPAN
- Switch virtualization
  - StackWise
    - StackWise master switch selection
    - Verifying StackWise
  - Virtual Switching System
    - Converting switches to a VSS
    - Verifying VSS

NOTE: If you are studying for the SWITCH certification exam, you might recognize that there are other topics in your studies that are usually part of this chapter. To maintain continuity, these topics have been moved to other chapters in this book:

- Configuring Network Time Protocol (NTP) is in Chapter 7, "Routers and Routing Protocol Hardening."
- Configuring Simple Network Management Protocol Version 3 (SNMPv3) is in Chapter 7.
- Configuring Basic IP SLAs is also in Chapter 5, "Path Control Implementation." There are more examples of IP SLAs shown in Chapter 13, "First-Hop Redundancy Implementation."

CAUTION: Your hardware platform or software release might not support all the commands documented in this chapter. Please refer to the Cisco website for specific platform and software release notes.

Configuring IP Service Level Agreements (Catalyst 3750)

Cisco IOS IP service level agreements (SLAs) send data across the network to measure performance between multiple network locations or network paths. They simulate network data and IP services and collect network performance information in real time. IP SLAs can also send SNMP traps that are triggered by events such as these:

- Connection loss
- Timeout
- Round-trip time threshold
- Average jitter threshold
- One-way packet loss
- One-way jitter
- One-way mean opinion score (MOS)
- One-way latency

Figure 12-1 is the network topology for the IP SLA commands.

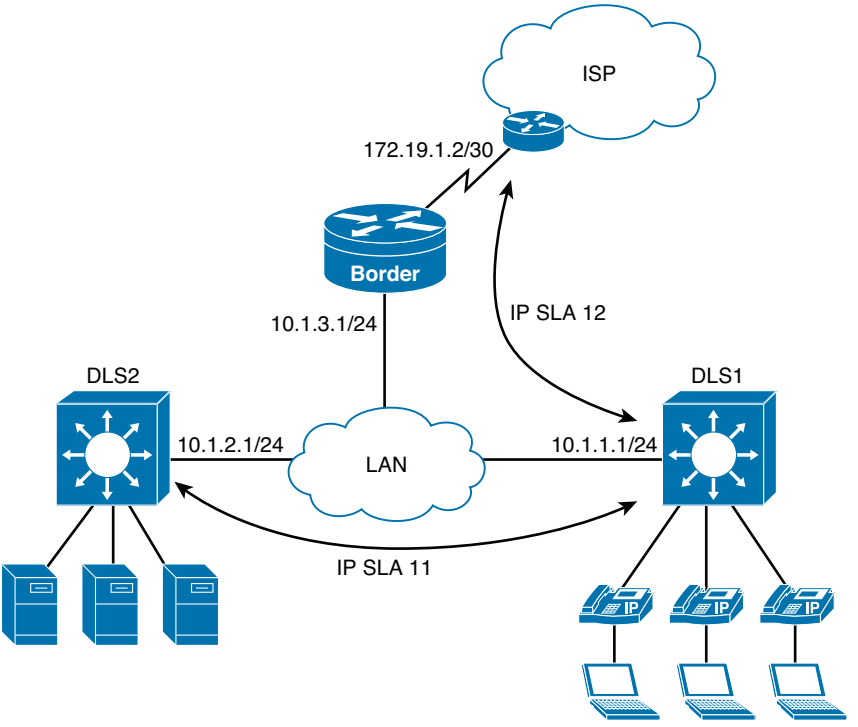


Figure 12-1 SLA Network Topology

`DLS1#configure terminal`	Enters global configuration mode.
`DLS1(config)#ip sla 11`	Creates an IP SLA operation and enters IP SLA configuration mode.
`DLS1(config-ip-sla)#icmp-echo 10.1.2.1 source-ip 10.1.1.1`	Configures the IP SLA operation as an ICMP echo operation and enters ICMP echo configuration mode.
	NOTE: The ICMP echo operation does not require the IP SLA responder to be enabled.
`DLS1(config-ip-sla-echo)# frequency 5`	Sets the rate at which the IP SLA operation repeats. Frequency is measured in seconds.
`DLS1(config-ip-sla-echo)#exit`	Exits IP SLA configuration mode.
`DLS1(config)#ip sla schedule 11 start-time now life forever`	Configures the IP SLA operation scheduling parameters to start now and continue forever.
	NOTE: The start time for the SLA can be set to a particular time and day, to be recurring, to be activated after a threshold is passed, and kept as an active process for a configurable number of seconds.
`DLS2(config)#ip sla responder`	Temporarily enables IP SLA responder functionality in response to control messages from the source.
`DLS1(config)#ip sla 12`	Creates an IP SLA operation and enters IP SLA configuration mode.
`DLS1(config-ip-sla)#path-jitter 172.19.1.2 source-ip 10.1.1.1`	Configures the IP SLA operation as a path-jitter operation and enters path-jitter configuration mode.
	NOTE: The path-jitter SLA sends 10 packets per operation with a 20-ms time interval between them by default.
`DLS1(config-ip-sla-pathJitter)# frequency 5`	Sets the rate at which the IP SLA operation repeats.
`DLS1(config-ip-sla-pathJitter)# tos 0x80`	Sets the type of service value to 0x80.
`DLS1(config-ip-sla-pathJitter)# exit`	Exits path-jitter configuration mode.
`DLS1(config)#ip sla schedule 12 recurring start-time 07:00 life 3600`	Configures the IP SLA operation scheduling parameters to start at 7 a.m. and continue for 1 hour every day.

Configuring Authentication for IP SLA

`Switch(config)#key chain Juliet`	Identifies a key chain.
`Switch(config-keychain)#key 1`	Identifies the key number.
`Switch(config-keychain)#key-string Shakespeare`	Identifies the key string.
`Switch(config-keychain)#exit`	Returns to global configuration mode.
`Switch(config)#ip sla key-chain Juliet`	Applies the key chain to the IP SLA process.
	NOTE: This must also be done on the responder.

Monitoring IP SLA Operations

`Switch#show ip sla application`	Displays global information about Cisco IOS IP SLAs.
	NOTE: The **show ip sla application** command displays supported SLA operation types and supported SLA protocols.
`Switch#show ip sla configuration 11`	Display configuration values including all defaults for SLA 11.
	NOTE: The use of a number in this command is optional.
`Switch#show ip sla statistics`	Display current or aggregated operational status and statistics.

Implementing Port Mirroring

Using a traffic sniffer can be a valuable tool to monitor and troubleshoot a network. In the modern era of switches, using the SPAN feature enables you to instruct a switch to send copies of packets seen on one port to another port on the same switch.

Default SPAN and RSPAN Configuration

The following table shows the default Switch Port Analyzer (SPAN) and Remote Switch Port Analyzer (RSPAN) configuration.

Feature	Default Setting
SPAN state (SPAN and RSPAN)	Disabled.
Source port traffic to monitor	Both received and sent traffic (both SPAN and RSPAN).
Encapsulation type (destination port)	Native form (untagged packets).
Ingress forwarding (destination port)	Disabled.

Feature	Default Setting
VLAN filtering	On a trunk interface used as a source port, all VLANs are monitored.
RSPAN VLANs	None configured.

Configuring Local SPAN

Local SPAN supports a SPAN session entirely within one switch; all source ports or source VLANs and destination ports are in the same switch or switch stack. Local SPAN copies traffic from one or more source ports in any VLAN or from one or more VLANs to a destination port for analysis.

Local SPAN Guidelines for Configuration

When configuring SPAN, follow these guidelines:

- For SPAN sources, you can monitor traffic for a single port or VLAN or a series or range of ports or VLANs for each session. You cannot mix source ports and source VLANs within a single SPAN session.
- The destination port cannot be a source port; a source port cannot be a destination port.
- You cannot have two SPAN sessions using the same destination port.
- When you configure a switch port as a SPAN destination port, it is no longer a normal switch port; only monitored traffic passes through the SPAN destination port.
- Entering SPAN configuration commands does not remove previously configured SPAN parameters. You must enter the **no monitor session** {*session_number* | **all** | **local** | **remote**} global configuration command to delete configured SPAN parameters.
- For local SPAN, outgoing packets through the SPAN destination port carry the original encapsulation headers (untagged or IEEE 802.1Q) if the **encapsulation replicate** *keywords* are specified. If the keywords are not specified, the packets are sent in native form. For RSPAN destination ports, outgoing packets are not tagged.
- You can configure a disabled port to be a source or destination port, but the SPAN function does not start until the destination port and at least one source port or source VLAN are enabled.
- You can limit SPAN traffic to specific VLANs by using the **filter vlan** keywords. If a trunk port is being monitored, only traffic on the VLANs specified with these keywords are monitored. By default, all VLANs are monitored on a trunk port.
- You cannot mix source VLANs and filter VLANs within a single SPAN session.

Configuring Local SPAN Example

Figure 12-2 is the network topology for Local SPAN commands.

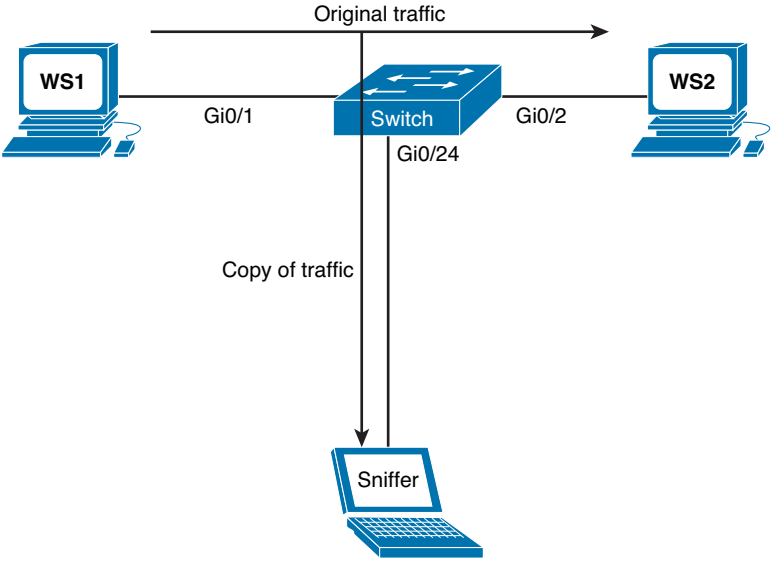


Figure 12-2 Local SPAN

`Switch(config)#no monitor session 1`	Removes any existing SPAN configuration on session 1. The session number is a number between 1 and 66.
`Switch(config)#no monitor session all`	Removes all SPAN sessions.
`Switch(config)#no monitor session local`	Removes all local SPAN sessions.
`Switch(config)#no monitor session remote`	Removes all remote SPAN sessions.
`Switch(config)#monitor session 1 source interface gigabitethernet 0/1`	Sets a new SPAN session where the source of the traffic will be interface Gigabit Ethernet 0/1.
`Switch(config)#monitor session 2 source gigabitethernet0/2 rx`	Configures session 2 to monitor received traffic on interface Gigabit Ethernet 0/2.

`Switch(config)#monitor session session_number source {interface interface-id | vlanvlan-id} [, | -] [both | rx | tx]`	Options for this command include the following: *session_number*: Any number between 1 and 66. *interface-id*: Specifies the source port to monitor. Can be any valid physical interface or port channel logical interface. *vlan-id*: Specifies the source VLAN to monitor. The range is 1 to 4094. , | **-** (optional): To be used to help specify a series or ranges of interfaces. There must be a space both before and after the comma or hyphen. **both** (optional): Monitors both received and sent traffic. This is the default setting. **rx** (optional): Monitors received traffic. **tx** (optional): Monitors sent traffic.
	NOTE: A single session can include multiple sources (ports or VLANs), defined in a series of commands, but you cannot combine source ports and source VLANs in one session.
	NOTE: You can use the **monitor session** *session_number* **source** command multiple times to configure multiple source ports.
`Switch(config)#monitor session 1 filter vlan 6 - 10`	Limits the SPAN source traffic to VLANs 6 to 10.
`Switch(config)#monitor session session_number filter vlan vlan-id [, | -]`	Options for this command include the following: *session_number*: Must match the session number used in the **monitor session source** command. *vlan-id*: Specifies the source VLAN to monitor. The range is 1 to 4094. , | **-** (optional): To be used to help specify a series or ranges of interfaces. There must be a space both before and after the comma or hyphen.
`Switch(config)#monitor session 1 destination interface gigabitethernet0/24 encapsulation replicate`	Sets a new SPAN session where the destination for the traffic will be interface Gigabit Ethernet 0/24. The encapsulation method will be retained.

`Switch(config)#monitor session 2 destination interface gigabitethernet0/24 encapsulation replicate ingress dot1q vlan 6`	Monitored traffic from session 2 will be sent to interface Gigabit Ethernet 0/24. It will have the same egress encapsulation type as the source port, and will enable ingress forwarding with IEEE 802.1Q encapsulation and VLAN 6 as the default ingress VLAN.
`Switch(config)#monitor session` *session_number* **destination** {**interface** *interface-id* [, | -] [**encapsulation {dot1q** | **replicate}]}** [**ingress** {**dot1q vlan** *vlan-id* | **untaggedvlan** *vlan-id* | **vlan** *vlan-id*}]}	Options for this command include the following: *session_number*: Enter in the session number used in the **source** command earlier in this example. For local SPAN, you *must* use the same session number for the source and destination interfaces. *interface-id*: Specifies the destination port. This must be a physical port; it cannot be an EtherChannel, and it cannot be a VLAN. , | **-** (optional): To be used to help specify a series or ranges of interfaces. There must be a space both before and after the comma or hyphen. **encapsulation dot1q**: Specifies that the destination interface use the IEEE 802.1Q encapsulation method. **encapsulation replicate**: Specifies that the destination interface replicates the source interface encapsulation method. **NOTE:** If no encapsulation method is selected, the default is to send packets in native form (untagged). **ingress dot1q vlan** *vlan-id*: Accept incoming packets with IEEE 802.1Q encapsulation with the specified VLAN as the default VLAN. **ingress untagged vlan** *vlan-id*: Accept incoming packets with untagged encapsulation with the specified VLAN as the default VLAN.**ingress vlan** *vlan-id*: Accept incoming packets with untagged encapsulation with the specified VLAN as the default VLAN.
	NOTE: You can use **monitor session** *session_number* **destination** command multiple times to configure multiple destination ports.

Configuring Remote SPAN

While local SPAN supports source and destination ports only on one switch, a Remote SPAN supports source and destination ports on different switches. RSPAN consists of an RSPAN VLAN, an RSPN source session, and an RSPAN destination session. You separately configure RSPAN source sessions and destination sessions on different switches.

Remote SPAN Guidelines for Configuration

When configuring RSPAN, follow these guidelines:

- All the items in the Local SPAN guidelines for configuration apply to RSPAN.
- Because RSPAN VLANs have special properties, you should reserve a few VLANs across your network for use as RSPAN VLANs; do not assign access ports to these VLANs.
- You can apply an output access control list (ACL) to RSPAN traffic to selectively filter or monitor specific packets. Specify these ACLs on the RSPAN VLAN in the RSPAN source switches.
- For RSPAN configuration, you can distribute the source ports and the destination ports across multiple switches in your network.
- RSPAN does not support bridge protocol data unit (BPDU) packet monitoring or other Layer 2 switch protocols.
- The RSPAN VLAN is configured only on trunk ports and not on access ports. To avoid unwanted traffic in RSPAN VLANs, make sure that the VLAN Remote SPAN feature is supported in all the participating switches.
- Access ports (including voice VLAN ports) on the RSPAN VLAN are put in the inactive state.
- RSPAN VLANs are included as sources for port-based RSPAN sessions when source trunk ports have active RSPAN VLANs. RSPAN VLANs can also be sources in SPAN sessions. However, because the switch does not monitor spanned traffic, it does not support egress spanning of packets on any RSPAN VLAN identified as the destination of an RSPAN source session on the switch.
- You can configure any VLAN as an RSPAN VLAN as long as these conditions are met:
    - The same RSPAN VLAN is used for an RSPAN session in all the switches.
    - All participating switches support RSPAN.
- We recommend that you configure an RSPAN VLAN before you configure an RSPAN source or a destination session.
- If you enable VTP and VTP pruning, RSPAN traffic is pruned in the trunks to prevent the unwanted flooding of RSPAN traffic across the network for VLAN IDs that are lower than 1005.

Configuring Remote SPAN Example

Figure 12-3 is the network topology for Remote SPAN commands.

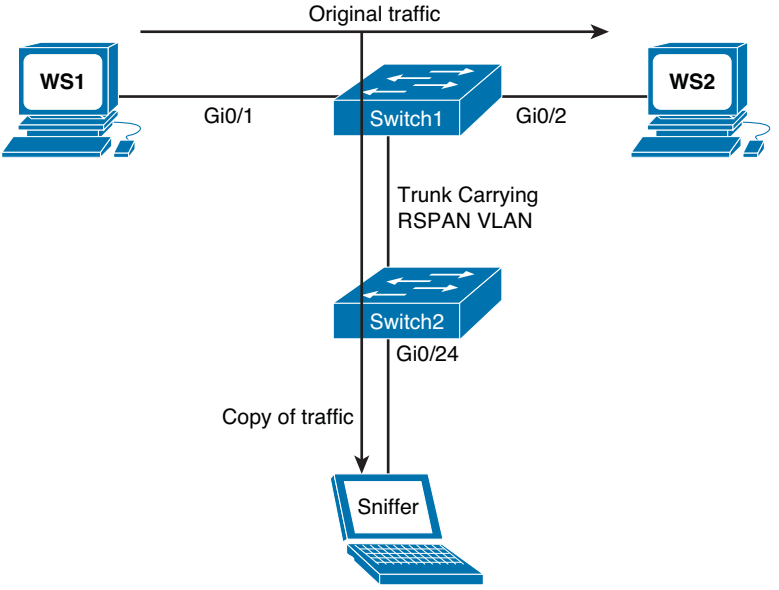


Figure 12-3 Remote SPAN

`Switch1(config)#vlan 901`	Creates VLAN 901 on Switch1
`Switch1(config-vlan)#remote span`	Makes this VLAN a RSPN VLAN
`Switch1(config-vlan)#end`	Returns to global configuration mode
`Switch2(config)#vlan 901`	Creates VLAN 901 on Switch2
`Switch2(config-vlan)#remote span`	Makes this VLAN a RSPN VLAN
`Switch2(config-vlan)#end`	Returns to global configuration mode

NOTE: You must create the RSPAN VLAN in all switches that will participate in RSPAN.

NOTE: If the RSPAN VLAN ID is in the normal range (lower than 1005) and VTP is enabled in the network, you can create the RSPAN VLAN in one switch, and VTP propagates it to the other switches in the VTP domain. For extended-range VLANs (greater than 1005), you must configure RSPAN VLAN on both source and destination switches and any intermediate switches.

TIP: Use VTP pruning to get an efficient flow of RSPAN traffic, or manually delete the RSPAN VLAN from all trunks that do not need to carry the RSPAN traffic.

`Switch1(config)#no monitor session 1`	Removes any previous configurations for session 1
`Switch1(config)#monitor session 1 source interface gigabitethernet0/1 tx`	Configures session 1 to monitor transmitted traffic on interface Gigabit Ethernet 0/1
`Switch1(config)#monitor session 1 source interface gigabitethernet0/2 rx`	Configures session 1 to monitor received traffic on interface Gigabit Ethernet 0/2
`Switch1(config)#monitor session 1 destination remote vlan 901`	Configures session 1 to have a destination of RSPAN VLAN 901
`Switch2(config)#no monitor session 1`	Removes any previous configurations for session 1
`Switch2(config)#monitor session 1 source remote vlan 901`	Configures session 1 to have a source of VLAN 901
`Switch2(config)#monitor session 1 destination interface gigabitethernet0/24`	Configures session 1 to have a destination interface of Gigabit Ethernet 0/24

NOTE: The commands to configure incoming traffic on a destination port and to filter VLAN traffic are the same on Remote SPAN as they are for Local SPAN.

Verifying and Troubleshooting Local and Remote SPAN

`Switch#show monitor session 1`	Displays output for SPAN session 1.
	NOTE: On some platforms the command is **show monitor**.
`Switch#show running-config`	Displays configuration of sessions running in active memory.
`Switch#show vlan remote-span`	Displays information about VLANs configured as RSPAN VLANs.
`Switch#debug monitor all`	Displays all SPAN debugging messages.
`Switch#debug monitor list`	Displays SPAN port and VLAN list tracing.
`Switch#debug monitor requests`	Displays SPAN requests.

Switch Virtualization

Redundant topologies often introduce overhead in terms of management, resiliency, and performance. To reduce the number of logical network devices and simplify Layer 2 and Layer 3 network topology, you can use two switch virtualization technologies: StackWise and Virtual Switching System (VSS).

StackWise

Cisco StackWise technology unites up to nine individual Cisco Catalyst 3750 switches or Cisco EtherSwitch service modules into a single logical unit, using special stack interconnect cables and stacking software. One of the Cisco EtherSwitch service modules or Catalyst 3750 switches controls the operation of the stack and is called the stack master. Switches can be added and deleted to a working stack without affecting stack performance.

NOTE: Catalyst 3750-E, 3750-X, and 3850 series switches support StackWise and StackWise Plus. StackWise Plus is an evolution of StackWise. StackWise Plus supports local switching, so locally destined packets need not traverse the stack ring.

NOTE: Catalyst 3850 series supports StackWise-480 with improved 480-Gbps stacking.

NOTE: Catalyst 2960-S series supports FlexStack, a StackWise-based feature tailored for Layer 2 switches. FlexStack is limited to four stacked switches.

NOTE: When a new switch is added, the stack master will automatically configure the unit with the configuration of the stack. The network manager does not have to do anything to bring up the switch before it is ready to operate.

StackWise Master Switch Selection

The hierarchy of selection criteria for the election of a master switch is as follows:

1. **User priority.** The network manager can select a switch to be master. This is done with the following global configuration command:

   ```
   Switch(config)#switch 1 priority 15
   ```

`Switch(config)#switch 1 priority 15`	Sets the priority of a switch in stack 1 to 15.
`Switch(config)#switch` *`stack-member-number`* `priority` *`new-priority-number`*	Options for this command include the following: *stack-member-number*: Specifies the current stack member number. The range is from 1 to 9. *new-priority-number*: Specifies the new stack member priority value. The range is from 1 to 15. The default is 1. A higher number increases the likelihood of a switch to be elected as stack master.
	NOTE: The new priority number is only a factor during a stack master reelection.

2. **Hardware and software priority.** This will default to the unit with the most extensive feature set. The Cisco Catalyst 3750 IP Services (IPS) image has the highest priority, followed by Cisco Catalyst 3750 switches with IP Base Software Image (IPB). Catalyst 3750-E and Catalyst 3750-X run the universal image. The feature set on the universal image is determined by the purchased license. The **show version** command will list operating license level for each switch member in the stack.
3. **Default configuration.** If a switch has preexisting configuration information, it will take precedence over switches that have not been configured.
4. **MAC address.** Each switch reports its MAC address to all its neighbors for comparison. The switch with the lowest MAC address is selected.

Verifying StackWise

`Switch#show platform stack manager all`	Displays all stack information
`Switch#show platform stack port buffer`	Displays the StackWise port events
`Switch#show platform stack port history`	Displays the StackWise history
`Switch#show switch`	Displays the shared MAC address and lists all switches in the stack with their stack number, role, MAC address, hardware priority, hardware version and current state
`Switch#show switch 1`	Displays information about stack member 1
`Switch#show switch detail`	Displays detailed information about the stack ring
`Switch#show switch neighbors`	Displays the stack neighbors
`Switch#show switch stack-ports`	Displays port information for the stack
`Switch#show switch stack-ports summary`	Displays a summary of the port information
`Switch#show switch stack-ring activity`	Displays the number of frames per member that are sent to the stack ring
`Switch#show switch stack-ring activity detail`	Displays the number of frames that are sent to the stack ring, the receive queue and the ASIC

Virtual Switching System

Virtual Switching System (VSS) is a network system virtualization technology that combines a pair of Catalyst 4500 or 6500 series switches into one virtual switch.

VSS is made up of two Catalyst switches and a Virtual Switch Link (VSL) between them. VSL is made up of up to eight 10-Gigabit Ethernet connections bundled into an EtherChannel. The VSL will carry both the control plane communication and the regular data traffic.

Converting Switches to a VSS

NOTE: When you convert two standalone switches into one VSS, all non-VSL configuration settings on the VSS standby chassis will revert to the default configuration.

The following steps are required when you convert two standalone chassis switches into a VSS.

Step 1: Back Up the Standalone Configurations

NOTE: This must be done on both switches.

Switch*X*#**copy running-config startup-config**	Saves the running configuration to startup configuration in NVRAM
Switch*X*#**copy start-up config disk0:old-startup-config**	Copies the startup configuration to a backup file

Step 2: Configure SSO and NSF

NOTE: Stateful switchover (SSO) and nonstop forwarding (NSF) are configured as default on the 4500s.

NOTE: This must be done on both switches.

6500Switch*X*(config)#**redundancy**	Enters into redundancy configuration mode.
6500Switch*X*(config-red)#**mode sso**	Configures SSO.
	NOTE: When this command is entered, the redundant supervisor engine is reloaded and begins to work in SSO mode.
6500Switch*X*(config-red)#**exit**	Returns to global configuration mode.
6500Switch*X*(config)#**router** *routing_protocol processID*	Enters into routing configuration mode.
6500Switch*X*(config-router)#**nsf**	Enables NSF operations for the routing protocol.
6500Switch*X*(config-router)#**end**	Returns to privileged mode.

Step 3: Assign Virtual Switch Domain and Switch Numbers

SwitchA(config)#**switch virtual domain 100**	Configures the virtual switch domain on Chassis A
SwitchA(config-vs-domain)#**switch 1**	Configures Chassis A as virtual switch number 1

`SwitchA(config-vs-domain)#`**`exit`**	Returns to global configuration mode
`SwitchB(config)#`**`switch virtual domain 100`**	Configures the virtual switch domain on Chassis B
`SwitchB(config-vs-domain)#`**`switch 2`**	Configures Chassis A as virtual switch number 2
`SwitchB(config-vs-domain)#`**`exit`**	Returns to global configuration mode

NOTE: The switch number is not stored in the startup or running configuration, because both chassis use the same configuration file (but must not have the same switch number).

NOTE: The domain number must be the same on both switches.

NOTE: One switch must be numbered switch 1, and the other switch must be numbered switch 2.

Step 4: Configure VSL Port Channel and Ports

NOTE: VSL is configured with a unique port channel on each chassis. Confirm that the port channel is available to use by issuing the **show running-config interface port-channel** *x* command. If the port channel is available, you will get an error message on the port channel number:

```
SwitchA#show running-config interface port-channel 10
                                                    ^
% Invalid input detected at '^' marker.
SwitchA#
```

`SwitchA(config)#`**`interface port-channel 10`**	Configures port channel 10 on SwitchA
`SwitchA(config-if)#`**`switch virtual link 1`**	Associates Switch 1 as owner of port channel 10
`SwitchA(config-if)#`**`no shutdown`**	Activates the port channel
`SwitchA(config-if)#`**`exit`**	Returns to global configuration mode
`SwitchA(config)#`**`interface range tengigabitethernet3/1-2`**	Enters configuration mode for interface range Ten Gigabit Ethernet 3/1-2
`SwitchA(config-if-range)#`**`channel-group 10 mode on`**	Adds these interfaces to channel group 10
`SwitchA(config-if-range)#`**`exit`**	Returns to global configuration mode
`SwitchB(config)#`**`interface port-channel 20`**	Configures port channel 20 on SwitchB
`SwitchB(config-if)#`**`switch virtual link 2`**	Associates Switch 2 as owner of port channel 20

`SwitchB(config-if)#no shutdown`	Activates the port channel
`SwitchB(config-if)#exit`	Returns to global configuration mode
`SwitchB(config)#interface range tengigabitethernet5/2-3`	Enters configuration mode for interface range Ten Gigabit Ethernet 5/2-3
`SwitchB(config-if-range)#channel-group 20 mode on`	Adds these interfaces to channel group 10
`SwitchB(config-if-range)#exit`	Returns to global configuration mode

TIP: For line redundancy, it is recommended to configure at least two ports per switch for the VSL. For module redundancy, the two ports can be on different switching modules in each chassis.

Step 5: Convert the Chassis to Virtual Switch Mode

Conversion to virtual switch mode requires a restart for both chassis.

NOTE: After the reboot, the chassis is in virtual switch mode, so commands that specify interfaces with module/port now include the switch number. For example, a port on a switching module is specified by switch/module/port.

`SwitchA#switch convert mode virtual`	Converts SwitchA to virtual switch mode. You will be prompted to confirm the action. Enter **yes**. At this point, the system will create a converted configuration file, and then saves the file to the RP bootflash.
`SwitchB#switch convert mode virtual`	Converts SwitchB to virtual switch mode. You will be prompted to confirm the action. Enter **yes**. At this point, the system will create a converted configuration file, and then saves the file to the RP bootflash.

NOTE: After you confirm the command (by entering **yes** at the prompt), the running configuration is automatically saved as the startup configuration and the chassis reboots.

Step 6: (Optional) Configure VSS Standby Chassis Modules

After the reboot, each chassis contains the module provisioning for its own slots. In addition, the modules from the VSS standby chassis have been automatically provisioned on the VSS active chassis with default configuration. In IOS versions earlier than IOS Release 12.2(50)SY, to provision modules on the VSS, use the **module provision** command in global configuration mode, as shown here.

`SwitchB(config)#module provision switch 2`	Enters into module provisioning configuration mode.
`SwitchB(config-prov-switch)#slot 3 slot-type 227 port-type 60 number 8 virtual slot 35`	Configures module provisioning: **slot 3** specifies the module number. **slot-type** is the VSL module type and the value 227 translates into the 8-port 10GE module. Valid values are 0–286. **port-type** of 60 indicates 10GE ports found on the 8-port 10GE module. The range is 1 to 100. **number 8** is the number of ports found on the actual module. **virtual-slot** *slot-num* specifies where the module fits in the switch. The keyword and argument is calculated as (Switch #* 16) + Slot #. In this case, 35 is calculated as 2 * 16 + 3 = 35.

NOTE: Do not delete or modify this section of the configuration file. In Cisco IOS Release 12.2(50)SY and later releases, you can no longer add module provisioning entries using the **module provision** command-line interface (CLI) command. When a module is not present, the provisioning entry for that module can be cleared using the **no slot** command with the **module provision** CLI command. Note that the VSS setup does not support the **module clear-config** command.

Verifying VSS

`Switch#show switch virtual`	Displays the virtual switch domain number, and the switch number and role for each of the chassis
`Switch#show switch virtual role`	Displays the role, switch number, and priority for each of the chassis in the VSS
`Switch#show switch virtual link`	Displays the status of the VSL
`Switch#show switch virtual link port-channel`	Displays more information about the VSL, such as EtherChannel used for the VSL
`Switch#show module provision switch`	Displays the module provisioning status

This page intentionally left blank

CHAPTER 13

First-Hop Redundancy Implementation

This chapter provides information about the following topics:

- First-hop redundancy
- Hot Standby Router Protocol
  - Configuring basic HSRP
  - Default HSRP configuration settings
  - Verifying HSRP
  - HSRP optimization options
    - Preempt
    - HSRP message timers
    - Authentication
    - Interface tracking
  - Multiple HSRP groups
  - HSRP IP SLA tracking
  - HSRPv2 for IPv6
  - Debugging HSRP
- Virtual Router Redundancy Protocol
  - Configuring VRRP
    - Interface tracking
  - Verifying VRRP
  - Debugging VRRP
- Gateway Load Balancing Protocol
  - Configuring GLBP
    - Interface tracking
  - Verifying GLBP
  - Debugging GLBP
- IPv4 configuration example: HSRP on L3 switch
- IPv4 configuration example: GLBP
- IPv4 configuration example: VRRP on Router and L3 Switch
- IPv6 configuration example: HSRPv2 on router and L3 switch

CAUTION: Your hardware platform or software release might not support all the commands documented in this chapter. Please refer to the Cisco website for specific platform and software release notes.

First-Hop Redundancy

A first-hop redundancy protocol (FHRP) is a networking protocol that is designed to protect the default gateway by allowing two or more routers or Layer 3 switches to provide backup for that address. If one first-hop device fails, the backup router will take over the address, by default, within a few seconds. FHRPs are equally at home on routers as Layer 3 (L3) switches. Hot Standby Router Protocol (HSRP), Virtual Router Redundancy Protocol (VRRP), and Gateway Load Balancing Protocol (GLBP) are implemented for both IPv4 and IPv6 environments. Platform IOS matrices should be consulted for next-hop redundancy protocol support.

Hot Standby Router Protocol

HSRP provides network redundancy for IP networks, ensuring that user traffic immediately and transparently recovers from first-hop failures in network-edge devices or access circuits.

When configuring HSRP on a switch platform, the specified interface must be a Layer 3 interface and Layer 3 functions enabled:

- **Routed port:** A physical port configured as a Layer 3 port by entering the **no switchport** interface configuration command.
- **SVI:** A VLAN interface created by using the **interface vlan** *vlan_id* global configuration command and by default a Layer 3 interface.
- **EtherChannel port channel in Layer 3 mode:** A port-channel logical interface created by using the **interface port-channel** *port-channel-number* global configuration command and binding the Ethernet interface into the channel group. For more information, see the "Configuring L3 EtherChannels" section in Chapter 9, "Campus Network Architecture."

Configuring Basic HSRP

`Switch(config)#interface vlan10`	Moves to interface configuration mode on the switch virtual interface (SVI).
`Switch(config-if)#ip address 172.16.0.10 255.255.255.0`	Assigns IP address and netmask.
`Switch(config-if)#standby 1 ip 172.16.0.1`	Activates HSRP group 1 on the interface and creates a virtual IP address of 172.16.0.1 for use in HSRP.
	NOTE: The group number can be from 0 to 255. The default is 0.

`Switch(config-if)#standby 1 priority 120`	Assigns a priority value of 120 to standby group 1.
	NOTE: The priority value can be from 1 to 255. The default is 100. A higher priority will result in that switch being elected the active switch. If the priorities of all switches in the group are equal, the switch with the **highest IP address** becomes the active switch.

NOTE: By and large, the HSRP configuration commands for a router are the same as HSRP on a Layer 3 switch platform.

Default HSRP Configuration Settings

Feature	Default Setting
HSRP version	Version 1
	NOTE: HSRPv1 and HSRPv2 have different packet structure. The same HSRP version must be configured on all devices of an HSRP group.
HSRP groups	None configured.
Standby group number	0
Standby MAC address	System assigned as 0000.0c07.ac*XX*, where *XX* is the HSRP group number. For HSRPv2, the MAC address will be 0000.0C9F.FXXX.
Standby priority	100
Standby delay	0 (no delay)
Standby track interface priority	10
Standby hello time	3 seconds
Standby holdtime	10 seconds

Verifying HSRP

`Switch#show standby`	Displays HSRP information
`Switch#show standby brief`	Displays a single-line output summary of each standby group
`Switch#show standby vlan 1`	Displays HSRP information on the VLAN 1 group

HSRP Optimization Options

Options are available that make it possible to optimize HSRP operation in the campus network. The next three sections explain four of these options: standby preempt, message timers, authentication, and interface tracking.

Preempt

`Switch(config)#interface vlan10`	Moves to interface configuration mode.
`Switch(config-if)#standby 1 preempt`	This switch will preempt, or take control of, the active switch if the local priority is higher than the priority of the active switch.
`Switch(config-if)#standby 1 preempt delay minimum 180 reload 140`	Causes the local switch to postpone taking over as the active switch for 180 seconds since that switch was last restarted or 140 seconds since the switch was last reloaded.
`Switch(config-if)#standby delay minimum 30 reload 60`	Sets a delay period for HSRP group initialization of 30 seconds when the interface comes up and 60 seconds after the switch reloads.
`Switch(config-if)#no standby 1 preempt delay`	Disables the preemption delay, but preemption itself is still enabled. Use the **no standby** *x* **preempt** command to eliminate preemption.
	NOTE: If the **preempt** argument is not configured, the local switch assumes control as the active switch only if the local switch receives information indicating that there is no switch currently in the active state.

HSRP Message Timers

`Switch(config)#interface vlan10`	Moves to interface configuration mode.
`Switch(config-if)#standby 1 timers 5 15`	Sets the hello timer to 5 seconds and sets the hold timer to 15 seconds.
	NOTE: The hold timer is normally set to be greater than or equal to 3 times the hello timer.
	NOTE: The hello timer can be from 1 to 254; the default is 3. The hold timer can be from 1 to 255; the default is 10. The default unit of time is seconds.
`Switch(config-if)#standby 1 timers msec 200 msec 600`	Sets the hello timer to 200 milliseconds and sets the hold timer to 600 milliseconds.
	NOTE: If the **msec** argument is used, the timers can be an integer from 15 to 999.

Authentication

`Switch(config)#key chain HSRP`	Creates an authentication key chain called **HSRP**.
`Switch(config-keychain)#key 1`	Adds a first key to the key chain.
`Switch(config-keychain-key)#key-string australia`	Configures a key string of **australia**.
`Switch(config)#interface vlan10`	Moves to interface configuration mode.
`Switch(config-if)#standby 1 authentication text canada`	Configures **canada** as the plain-text authentication string used by group 1.
`Switch(config-if)#standby 2 authentication md5 key-string england`	Configures **england** as the MD5 authentication key string used by group 2.
`Switch(config-if)#standby 3 authentication md5 key-chain HSRP`	Configures MD5 authentication using key chain **HSRP**. HSRP queries the key chain to obtain the current live key and key ID.

Interface Tracking

`Switch(config)#interface vlan10`	Moves to interface configuration mode.
`Switch(config-if)#standby 1 track fastethernet0/0 25`	HSRP will track the availability of interface FastEthernet0/0. If FastEthernet0/0 goes down, the priority of the switch in group 1 will be decremented by 25.
	NOTE: The default value of the **track** argument is 10.
	TIP: The **track** argument does not assign a new priority if the tracked interface goes down. The **track** argument assigns a value that the priority will be decreased if the tracked interface goes down. Therefore, if you are tracking FastEthernet0/0 with a track value of 25 (**standby 1 track fastethernet 0/0 25**) and FastEthernet0/0 goes down, the priority will be decreased by 25; assuming a default priority of 100, the new priority will now be 75.

Multiple HSRP Groups

Figure 13-1 shows the network topology for the configuration that follows, which demonstrates how to configure multiple HSRP groups using the commands covered in this chapter. Note that only the commands specific to HSRP and STP are shown in this example.

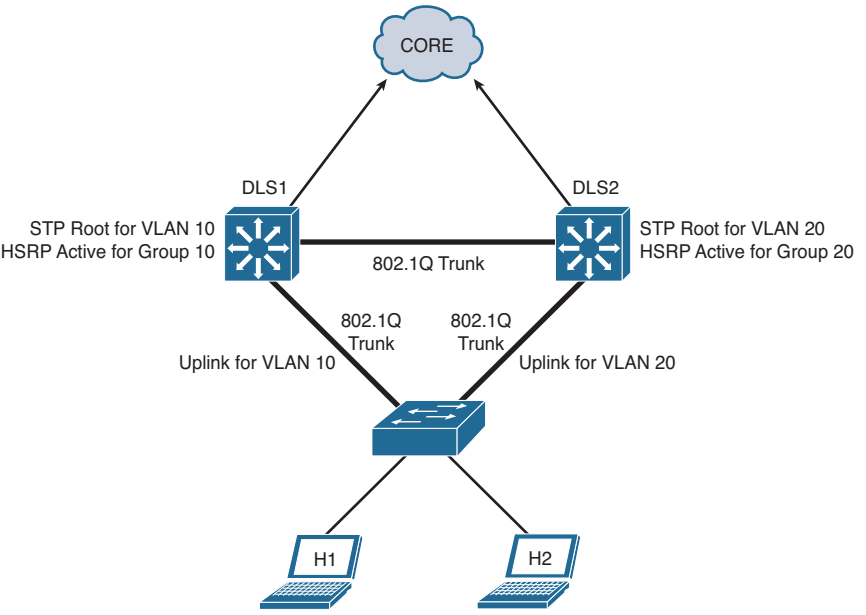


Figure 13-1 Network Topology for Multigroup HSRP Configuration Example

Multigroup HSRP enables switches to simultaneously provide redundant backup and perform load sharing across different IP subnets. The objective here is to configure DLS1 as STP root and HSRP active for VLAN 10, while DLS2 is configured as STP root and HSRP active for VLAN 20. DLS1 is also configured as backup root and HSRP standby for VLAN 20, while DLS2 is configured as backup root and HSRP standby for VLAN 10. Only the configuration for DLS1 is shown here. DLS2 would be configured in the opposite way.

`DLS1(config)#spanning-tree vlan 10 root primary`	Configures spanning-tree root primary for VLAN 10.
`DLS1(config)#spanning-tree vlan 20 root secondary`	Configures spanning-tree root secondary for VLAN 20.
	NOTE: Load balancing can be accomplished by having one switch be the active HSRP L3 switch forwarding for half of the VLANs and the standby L3 switch for the remaining VLANs. The second HSRP L3 switch would be reversed in its active and standby VLANs. Care must be taken to ensure that spanning-tree is forwarding to the active L3 switch for the correct VLANs by making that L3 switch the spanning-tree primary root for those VLANs.

`DLS1(config)#interface vlan10`	Moves to interface configuration mode.
`DLS1(config-if)#ip address 10.1.10.2 255.255.255.0`	Assigns IP address and netmask.
`DLS1(config-if)#standby 10 ip 10.1.10.1`	Activates HSRP group 10 on the interface and creates a virtual IP address of 10.1.10.1 for use in HSRP.
`DLS1(config-if)#standby 10 priority 110`	Assigns a priority value of 110 to standby group 10. This will be the active forwarded for VLAN 10.
`DLS1(config-if)#standby 10 preempt`	This switch will preempt, or take control of, VLAN 10 forwarding if the local priority is higher than the active switch VLAN 10 priority.
`DLS1(config-if)#interface vlan20`	Moves to interface configuration mode.
`DLS1(config-if)#ip address 10.1.20.2 255.255.255.0`	Assigns IP address and netmask.
`DLS1(config-if)#standby 20 ip 10.1.20.1`	Activates HSRP group 20 on the interface and creates a virtual IP address of 10.1.20.1 for use in HSRP.
`DLS1(config-if)#standby 20 priority 90`	Assigns a priority value of 90 to standby group 20. This switch will be the standby device for VLAN 20.
`DLS1(config-if)#standby 20 preempt`	This switch will preempt, or take control of, VLAN 20 forwarding if the local priority is higher than the active switch VLAN 20 priority.

HSRP IP SLA Tracking

See Chapter 5, "Path Control Implementation," for a more detailed explanation of IP service level agreement (SLA) objects. The objective here is to associate an IP SLA to the HSRP process, allowing failover to occur by decrementing the HSRP priority if the object fails.

`Switch(config)#ip sla 10`	Creates SLA process 10.
`Switch(config-sla)#icmp-echo 172.19.10.1`	Configures the SLA as an ICMP Echo operation to destination 172.19.10.1.
`Switch(config-sla)#exit`	Exits SLA configuration mode.
`Switch(config)#ip sla schedule 10 start-time now life forever`	Configures the scheduling for SLA 10 to start now and continue forever.
`Switch(config)#track 90 ip sla 10 state`	Creates an object, 90, to track the state of SLA process 10.
`Switch(config)#interface vlan10`	Moves to interface configuration mode.

`Switch(config-if)#ip address 192.168.10.1 255.255.255.0`	Assigns IP address and netmask.
`Switch(config-if)#standby 10 ip 192.168.10.254`	Activates HSRP group 10 on the interface and creates a virtual IP address of 192.168.10.254 for use in HSRP.
`Switch(config-if)#standby 10 priority 110`	Assigns a priority value of 110 to standby group 10.
`Switch(config-if)#standby 10 preempt`	This switch will preempt, or take control of, the active switch if the local priority is higher than the active switch.
`Switch(config-if)#standby 10 track 90 decrement 20`	Track the state of object 90 and decrement the device priority if the object fails.

HSRPv2 for IPv6

HSRP Version 2 must be enabled on an interface before HSRP for IPv6 can be configured.

`Switch(config-if)#standby version 2`	Enables HSRPv2 on an interface
`Switch(config-if)#standby 1 ipv6 autoconfig`	Enables HSRP for IPv6 using a virtual link-local address that will be generated automatically from the link-local prefix and a modified EUI-64 format interface identifier, where the EUI-64 interface identifier is created from the relevant HSRP virtual MAC address
`Switch(config-if)#standby 1 ipv6 FE80::1:1`	Enables HSRP for IPv6 using an explicitly configured link-local address to be used as the virtual IPv6 address for group 1
`Switch(config-if)#standby 1 ipv6 2001::0DB8:2/64`	Enables HSRP for IPv6 using a global IPv6 address as the virtual address for group 1

NOTE: All other relevant HSRP commands (preempt, priority, authentication, tracking, and so on) are identical in HSRPv1 and HSRPv2.

NOTE: When configuring the IPv6 virtual address, if an IPv6 global address is used, it must include an IPv6 prefix length. If a link-local address is used, it does not have a prefix.

Debugging HSRP

`Switch#debug standby`	Displays all HSRP debugging information, including state changes and transmission/reception of HSRP packets
`Switch#debug standby errors`	Displays HSRP error messages
`Switch#debug standby events`	Displays HSRP event messages
`Switch#debug standby events terse`	Displays all HSRP events except for hellos and advertisements
`Switch#debug standby events track`	Displays all HSRP tracking events
`Switch#debug standby packets`	Displays HSRP packet messages
`Switch#debug standby terse`	Displays all HSRP errors, events, and packets, except for hellos and advertisements

Virtual Router Redundancy Protocol

NOTE: HSRP is Cisco proprietary. The Virtual Router Redundancy Protocol (VRRP) is an IEEE standard.

NOTE: VRRP might not be completely supported on platforms such as the Catalyst 3750-E, 3750, 3560, or 3550. For example, the Catalyst 3560 supports VRRP for IPv4, but not for IPv6. The IPv4 implementation supports text authentication, but not message digest 5 (MD5) authentication key chain implementation. Also, the Switch Database Management (SDM) should prefer the **routing** option for IPv4 or the **dual-ipv4-and-ipv6** option for dual-stack or IPv6 implementations. VRRP is supported on the Catalyst 4500 and Catalyst 6500 platforms. Verify VRRP capabilities by platform datasheets and appropriate Cisco IOS command and configuration guides.

VRRP is an election protocol that dynamically assigns responsibility for one or more virtual switches to the VRRP switches on a LAN, allowing several switches on a multiaccess link to use the same virtual IP address. A VRRP switch is configured to run VRRP in conjunction with one or more other switches attached.

Configuring VRRP

`Switch(config)#interface vlan10`	Moves to interface configuration mode.
`Switch(config-if)#ip address 172.16.100.5 255.255.255.0`	Assigns IP address and netmask.
`Switch(config-if)#vrrp 10 ip 172.16.100.1`	Enables VRRP for group 10 on this interface with a virtual address of 172.16.100.1. The group number can be from 1 to 255.
	NOTE: VRRP supports using the real interface IP address as the virtual IP for the group. If this is done, the router with that address becomes the master.

`Switch(config-if)#vrrp 10 description Engineering Group`	Assigns a text description to the group.
`Switch(config-if)#vrrp 10 priority 110`	Sets the priority level for this VLAN. The range is from 1 to 254. The default is 100.
`Switch(config-if)#vrrp 10 preempt`	This switch will preempt, or take over, as the virtual switch master for group 10 if it has a higher priority than the current virtual switch master.
	NOTE: The switch that is the IP address owner will preempt, regardless of the setting of this command.
	NOTE: The **preempt** VRRP option is enabled by default.
`Switch(config-if)#vrrp 10 preempt delay minimum 60`	This switch will preempt, but only after a delay of 60 seconds.
	NOTE: The default delay period is 0 seconds.
`Switch(config-if)#vrrp 10 timers advertise 15`	Configures the interval between successful advertisements by the virtual switch master.
	NOTE: The default interval value is 1 second.
	NOTE: All switches in a VRRP group must use the same timer values. If switches have different timer values set, the VRRP group will not communicate with each other.
	NOTE: The range of the advertisement timer is 1 to 255 seconds. If you use the **msec** argument, you change the timer to measure in milliseconds. The range in milliseconds is 50 to 999.
`Switch(config-if)#vrrp 10 timers learn`	Configures the switch, when acting as a virtual switch backup, to learn the advertisement interval used by the virtual switch master.
`Switch(config-if)#vrrp 10 shutdown`	Disables VRRP on the interface, but configuration is still retained.
`Switch(config-if)#no vrrp 10 shutdown`	Reenables the VRRP group using the previous configuration.
`Switch(config-if)vrrp 10 authentication text ottawa`	Configures plain-text authentication for group 10 using the key **ottawa**.
`Switch(config-if)#vrrp 10 authentication md5 key-string winnipeg`	Configures MD5 authentication for group 10 using the key **winnipeg**.

Interface Tracking

VRRP does not have a native interface tracking mechanism. Instead, it has the ability to track objects. This allows the VRRP master to lose its status if a tracked object (interface, IP SLA, and so on) fails.

`Switch(config)#track 10 interface fastethernet0/0 line-protocol`	Creates a tracked object, where the status of the uplink interface is tracked
`Switch(config)#interface fastethernet0/1`	Moves to interface configuration mode
`Switch(config)#vrrp 1 track 10 decrement 30`	Configures VRRP to track the previously created object and decrease the VRRP priority by 30 should the uplink interface fail

Verifying VRRP

NOTE: The VRRP verification commands are the same for IPv6 and IPv4.

`Switch#show vrrp`	Displays VRRP information
`Switch#show vrrp brief`	Displays a brief status of all VRRP groups
`Switch#show vrrp 10`	Displays detailed information about VRRP group 10
`Switch#show vrrp interface vlan10`	Displays information about VRRP as enabled on interface Vlan10
`Switch#show vrrp interface vlan10 brief`	Displays a brief summary about VRRP on interface Vlan10

Debugging VRRP

`Switch#debug vrrp all`	Displays all VRRP messages
`Switch#debug vrrp error`	Displays all VRRP error messages
`Switch#debug vrrp events`	Displays all VRRP event messages
`Switch#debug vrrp packets`	Displays messages about packets sent and received
`Switch#debug vrrp state`	Displays messages about state transitions

Gateway Load Balancing Protocol

Gateway Load Balancing Protocol (GLBP) protects data traffic from a failed router or circuit, such as HSRP and VRRP, while allowing packet load sharing between a group of redundant routers. Like HSRP, it is Cisco proprietary.

Configuring GLBP

`Router(config)#interface fastethernet0/0`	Moves to interface configuration mode.
`Router(config-if)#ip address 172.16.100.5 255.255.255.0`	Assigns IP address and netmask.
`Router(config-if)#glbp 10 ip 172.16.100.1`	Enables GLBP for group 10 on this interface with a virtual address of 172.16.100.1. The range of group numbers is from 0 to 1023.
`Router(config-if)#glbp 10 preempt`	Configures the router to preempt, or take over, as the active virtual gateway (AVG) for group 10 if this router has a higher priority than the current AVG. Preemption is disabled by default.
`Router(config-if)#glbp 10 preempt delay minimum 60`	Configures the router to preempt, or take over, as AVG for group 10 if this router has a higher priority than the current AVG after a delay of 60 seconds
`Router(config-if)#glbp 10 forwarder preempt`	Configures the router to preempt, or take over, as AVF for group 10 if this router has a higher priority than the current AVF. This command is enabled by default with a delay of 30 seconds.
`Router(config-if)#glbp 10 forwarder preempt delay minimum 60`	Configures the router to preempt, or take over, as AVF for group 10 if this router has a higher priority than the current AVF after a delay of 60 seconds.
	NOTE: Members of a GLBP group elect one gateway to be the AVG for that group. Other group members provide backup for the AVG in the event that the AVG becomes unavailable. The AVG assigns a virtual MAC address to each member of the GLBP group. Each gateway assumes responsibility for forwarding packets sent to the virtual MAC address assigned to it by the AVG. These gateways are known as AVFs for their virtual MAC address.Virtual forwarder redundancy is similar to virtual gateway redundancy with an AVF. If the AVF fails, one of the secondary virtual forwarders in the listen state assumes responsibility for the virtual MAC address.
`Router(config-if)#glbp 10 priority 150`	Sets the priority level of the router.
	NOTE: The range of the **priority** argument is 1 to 255. The default priority of GLBP is 100. A higher priority number is preferred.

`Router(config-if)#glbp 10 timers 5 15`	Configures the hello timer to be set to 5 seconds and the hold timer to be 15 seconds.
`Router(config-if)#glbp 10 timers msec 20200 msec 60600`	Configures the hello timer to be 20,200 milliseconds and the hold timer to be 60,600 milliseconds.
	NOTE: The default hello timer is 3 seconds. The range of the hello timer interval is 1 to 60 seconds. If the **msec** argument is used, the timer will be measured in milliseconds, with a range of 50 to 60,000.
	NOTE: The default hold timer is 10 seconds. The range of the hold timer is 19 to 180 seconds. If the **msec** argument is used, the timer will be measured in milliseconds, with a range of 18,020 to 180,000.The hello timer measures the interval between successive hello packets sent by the AVG in a GLBP group. The **holdtime** argument specifies the interval before the virtual gateway and the virtual forwarder information in the hello packet is considered invalid. It is recommended that unless you are extremely familiar with your network design and with the mechanisms of GLBP that you do not change the timers. To reset the timers back to their default values, use the **no glbp x timers** command, where **x** is the GLBP group number.
`Router(config)#glbp 10 authentication text edmonton`	Configures GLBP for plain text authentication of group 10 GLBP packets with a key of **edmonton**.
`Router(config)#glbp 10 authentication md5 key-chain vancouver`	Configures GLBP for MD5 authentication of group 10 GLBP packets with a key of **vancouver**.
`Router(config-if)#glbp 10 load-balancing host-dependent`	Specifies that GLBP will load balance using the host-dependent method.
`Router(config-if)#glbp 10 load-balancing weighted`	Specifies that GLBP will load balance using the weighted method.
`Router(config-if)#glbp 10 weighting 80`	Assigns a maximum weighting value for this interface for load-balancing purposes. The value can be from 1 to 254.
`Router(config-if)#glbp 10 load balancing round robin`	Specifies that GLBP will load balance using the round-robin method.

NOTE: There are three different types of load balancing in GLBP:

- **Host-dependent** uses the MAC address of a host to determine which VF MAC address the host is directed toward. This is used with stateful Network Address Translation (NAT) because NAT requires each host to be returned to the same virtual MAC address each time it sends an ARP request for the virtual IP address. It is not recommended for situations where there are a small number of end hosts (fewer than 20).
- **Weighted** allows for GLBP to place a weight on each device when calculating the amount of load sharing. For example, if there are two routers in the group, and router A has twice the forwarding capacity of router B, the weighting value should be configured to be double the amount of router B. To assign a weighting value, use the **glbp x weighting y** interface configuration command, where **x** is the GLBP group number, and **y** is the weighting value, a number from 1 to 254.
- **Round-robin** load balancing occurs when each VF MAC address is used sequentially in ARP replies for the virtual IP address. Round-robin is suitable for any number of end hosts.If no load balancing is used with GLBP, GLBP will operate in an identical manner to HSRP, where the AVG will only respond to ARP requests with its own VF MAC address, and all traffic will be directed to the AVG. The command to achieve this is **no glbp load-balancing**.

Interface Tracking

`Router(config)#track 2 interface fastethernet0/1 line-protocol`	Configures the FastEthernet0/1 interface to be tracked. The **line-protocol** keyword tracks whether the interface is up
`Router(config-track)#exit`	Returns to global configuration mode
`Router(config)#interface fastethernet0/0`	Enters interface configuration mode
`Router(config-if)#glbp 10 weighting 110 lower 20 upper 50`	Specifies the initial weighting value, and the upper and lower thresholds, for a GLBP gateway
`Router(config-if)#glbp 10 weighting track 2 decrement 50`	Tells GLBP to track the object and decrement the weight by 50 when the Fast Ethernet 0/1 interface fails

Verifying GLBP

`Router#show glbp`	Displays GLBP information
`Router#show glbp brief`	Displays a brief status of all GLBP groups
`Router#show glbp 10`	Displays information about GLBP group 10
`Router#show glbp vlan10`	Displays GLBP information on interface Vlan10
`Router#show glbp vlan20 10`	Displays GLBP group 10 information on interface Vlan20

Debugging GLBP

`Router#debug condition glbp`	Displays GLBP condition messages
`Router#debug glbp errors`	Displays all GLBP error messages
`Router#debug glbp events`	Displays all GLBP event messages
`Router#debug glbp packets`	Displays messages about packets sent and received
`Router#debug glbp terse`	Displays a limited range of debugging messages

IPv4 Configuration Example: HSRP on L3 Switch

Figure 13-2 shows the network topology for the configuration that follows, which demonstrates how to configure HSRP using the commands covered in this chapter. Note that only the commands specific to HSRP are shown in this example.

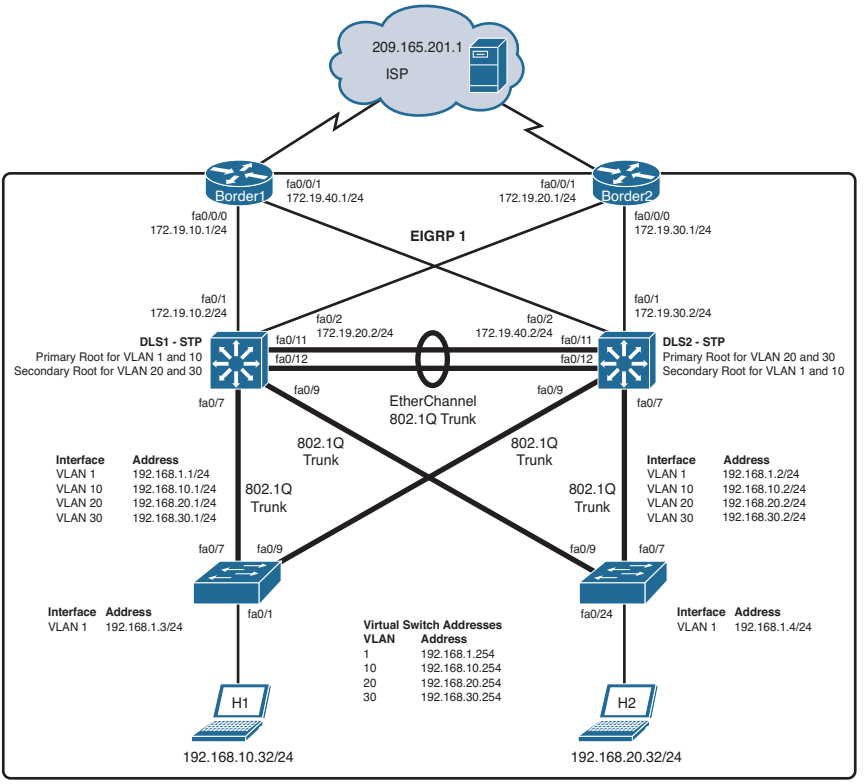


Figure 13-2 Network Topology for HSRP Configuration Example

The network devices are configured:

- DLS1 and DLS2 are configured as Layer 3 devices; ALS1 and ALS2 are configured as Layer 2 devices.
- Border1, Border2, DLS1, and DLS2 run Enhanced Interior Gateway Routing Protocol (EIGRP). Border1 and Border2 also provide default routing into the cloud.
- The links between DLS1, DLS2, Border1, and Border2 are routed links using the **no switchport** command on DLS1 and DLS2.
- Four VLANs are configured on DLS1. DLS1 is the VTP server for DLS2, ALS1, and ALS2.
- A Layer 2 EtherChannel connects DLS1 and DLS2.
- All connections between DLS1, DLS2, ALS1, and ALS2 are 802.1Q trunks.
- DLS1 is the spanning-tree primary root for VLAN 1 and 10 and DLS1 is the secondary root for VLAN 20 and 30.
- DLS2 is the spanning-tree primary root for VLAN 20 and 30 and DLS1 is the secondary root for VLAN 1 and 10.
- DLS1 is to be HSRP active for VLAN 1 and 10, and HSRP standby for VLAN 20 and 30.
- DLS2 is to be HSRP active for VLAN 20 and 30, and HSRP standby for VLAN 1 and 10.
- Interface tracking is configured to allow for HSRP failover to occur if an uplink fails.

Switch DLS1

`DLS1(config)#interface vlan1`	Moves to interface configuration mode.
`DLS1(config-if)#standby 1 ip 192.168.1.254`	Activates HSRP group 1 on the interface and creates a virtual IP address of 192.168.1.254 for use in HSRP.
`DLS1(config-if)#standby 1 priority 105`	Assigns a priority value of 105 to standby group 1.
`DLS1(config-if)#standby 1 preempt`	This switch will preempt, or take control of, vlan 1 forwarding if the local priority is higher than the active switch VLAN 1 priority.
`DLS1(config-if)#standby 1 track fastethernet0/1 20`	HSRP will track the availability of interface FastEthernet0/1. If FastEthernet0/1 goes down, the priority of the switch in group 1 will be decremented by 20.

DLS1(config-if)#**standby 1 track fastethernet0/2**	HSRP will track the availability of interface FastEthernet0/2. If FastEthernet0/2 goes down, the priority of the switch in group 1 will be decremented by the default value of 10.
DLS1(config-if)#**exit**	Moves to global configuration mode.
DLS1(config)#**interface vlan10**	Moves to interface configuration mode.
DLS1(config-if)#**standby 10 ip 192.168.10.254**	Activates HSRP group 10 on the interface and creates a virtual IP address of 192.168.10.254 for use in HSRP.
DLS1(config-if)#**standby 10 priority 105**	Assigns a priority value of 105 to standby group 10.
DLS1(config-if)#**standby 10 preempt**	This switch will preempt, or take control of, VLAN 10 forwarding if the local priority is higher than the active switch VLAN 10 priority.
DLS1(config-if)#**standby 10 track fastethernet0/1 20**	HSRP will track the availability of interface FastEthernet0/1. If FastEthernet0/1 goes down, the priority of the switch in group 10 will be decremented by 20.
DLS1(config-if)#**standby 10 track fastethernet0/2**	HSRP will track the availability of interface FastEthernet0/2. If FastEthernet0/2 goes down, the priority of the switch in group 10 will be decremented by the default value of 10.
DLS1(config-if)#**exit**	Moves to global configuration mode.
DLS1(config)#**interface vlan20**	Moves to interface configuration mode.
DLS1(config-if)#**standby 20 ip 192.168.20.254**	Activates HSRP group 20 on the interface and creates a virtual IP address of 192.168.20.254 for use in HSRP.
DLS1(config-if)#**standby 20 priority 100**	Assigns a priority value of 100 to standby group 20.
DLS1(config-if)#**standby 20 track fastethernet0/1 20**	HSRP will track the availability of interface FastEthernet0/1. If FastEthernet0/1 goes down, the priority of the switch in group 20 will be decremented by 20.
DLS1(config-if)#**standby 20 track fastethernet0/2**	HSRP will track the availability of interface FastEthernet0/2. If FastEthernet0/2 goes down, the priority of the switch in group 20 will be decremented by the default value of 10.
DLS1(config-if)#**exit**	Moves to global configuration mode.
DLS1(config)#**interface vlan30**	Moves to interface configuration mode.

DLS1(config-if)#**standby 30 ip 192.168.30.254**	Activates HSRP group 30 on the interface and creates a virtual IP address of 192.168.30.254 for use in HSRP.
DLS1(config-if)#**standby 30 priority 100**	Assigns a priority value of 100 to standby group 30.
DLS1(config-if)#**standby 30 track fastethernet0/1 20**	HSRP will track the availability of interface FastEthernet0/1. If FastEthernet0/1 goes down, the priority of the switch in group 30 will be decremented by 20.
DLS1(config-if)#**standby 30 track fastethernet0/2**	HSRP will track the availability of interface FastEthernet0/2. If FastEthernet0/2 goes down, the priority of the switch in group 30 will be decremented by the default value of 10.
DLS1(config-if)#**exit**	Moves to global configuration mode.

Switch DLS2

DLS2(config)#**interface vlan1**	Moves to interface configuration mode.
DLS2(config-if)#**standby 1 ip 192.168.1.254**	Activates HSRP group 1 on the interface and creates a virtual IP address of 192.168.1.254 for use in HSRP.
DLS2(config-if)#**standby 1 priority 100**	Assigns a priority value of 100 to standby group 1.
DLS2(config-if)#**standby 1 track fastethernet0/1 20**	HSRP will track the availability of interface FastEthernet0/1. If FastEthernet0/1 goes down, the priority of the switch in group 1 will be decremented by 20.
DLS2(config-if)#**standby 1 track fastethernet0/2**	HSRP will track the availability of interface FastEthernet0/2. If FastEthernet0/2 goes down, the priority of the switch in group 1 will be decremented by the default value of 10.
DLS2(config-if)#**exit**	Moves to global configuration mode.
DLS2(config)#**interface vlan10**	Moves to interface configuration mode.
DLS2(config-if)#**standby 10 ip 192.168.10.254**	Activates HSRP group 10 on the interface and creates a virtual IP address of 192.168.10.254 for use in HSRP.
DLS2(config-if)#**standby 10 priority 100**	Assigns a priority value of 100 to standby group 10.

DLS2(config-if)#**standby 10 track fastethernet0/1 20**	HSRP will track the availability of interface FastEthernet0/1. If FastEthernet0/1 goes down, the priority of the switch in group 10 will be decremented by 20.
DLS2(config-if)#**standby 10 track fastethernet0/2**	HSRP will track the availability of interface FastEthernet0/2. If FastEthernet0/2 goes down, the priority of the switch in group 10 will be decremented by the default value of 10.
DLS2(config-if)#**exit**	Moves to global configuration mode.
DLS2(config)#**interface vlan20**	Moves to interface configuration mode.
DLS2(config-if)#**standby 20 ip 192.168.20.254**	Activates HSRP group 20 on the interface and creates a virtual IP address of 192.168.20.254 for use in HSRP.
DLS2(config-if)#**standby 20 priority 105**	Assigns a priority value of 105 to standby group 20.
DLS2(config-if)#**standby 20 preempt**	This switch will preempt, or take control of, VLAN 20 forwarding if the local priority is higher than the active switch VLAN 20 priority.
DLS2(config-if)#**standby 20 track fastethernet0/1 20**	HSRP will track the availability of interface FastEthernet0/1. If FastEthernet0/1 goes down, the priority of the switch in group 20 will be decremented by 20.
DLS2(config-if)#**standby 20 track fastethernet0/2**	HSRP will track the availability of interface FastEthernet0/2. If FastEthernet0/2 goes down, the priority of the switch in group 20 will be decremented by the default value of 10.
DLS2(config-if)#**exit**	Moves to global configuration mode.
DLS2(config)#**interface vlan30**	Moves to interface configuration mode.
DLS2(config-if)#**standby 30 ip 192.168.30.254**	Activates HSRP group 30 on the interface and creates a virtual IP address of 192.168.30.254 for use in HSRP.
DLS2(config-if)#**standby 30 priority 105**	Assigns a priority value of 105 to standby group 30.
DLS2(config-if)#**standby 30 preempt**	This switch will preempt, or take control of, VLAN 30 forwarding if the local priority is higher than the active switch VLAN 30 priority.

`DLS2(config-if)#standby 30 track fastethernet0/1 20`	HSRP will track the availability of interface FastEthernet0/1. If FastEthernet0/1 goes down, the priority of the switch in group 30 will be decremented by 20.
`DLS2(config-if)#standby 30 track fastethernet0/2`	HSRP will track the availability of interface FastEthernet0/2. If FastEthernet0/2 goes down, the priority of the switch in group 30 will be decremented by the default value of 10.
`DLS2(config-if)#exit`	Moves to global configuration mode.

IP SLA Tracking: Switch DLS1 VLAN 10

Refer to Figure 13-2. The objective here is to probe the availability of a web server hosted in the ISP cloud at address 209.165.201.1. If the server does not respond to the IP SLA ping, the HSRP priority on interface VLAN 10 will be decremented by 20. This configuration could be applied to all other VLANs where the HSRP Active device resides (DLS1 for VLAN 1 and 10; DLS2 for VLAN 20 and 30).

`DLS1(config)#ip sla 10`	Creates SLA process 10
`DLS1(config-ip-sla)#icmp-echo 192.168.10.1`	Configures the SLA as an ICMP echo operation to destination 192.168.10.1
`DLS1(config-ip-sla-echo)#exit`	Exits SLA configuration mode
`DLS1(config)#ip sla schedule 10 start-time now life forever`	Configures the scheduling for SLA 10 process to start now and continue forever
`DLS1(config)#track 90 ip sla 10 state`	Creates an object, 90, to track the state of SLA process 10
`DLS1(config-track)#exit`	Moves to global configuration mode
`DLS1(config)#interface vlan10`	Moves to interface configuration mode
`DLS1(config-if)#standby 10 track 90 decrement 20`	Tracks the state of object 90 and decrement the device priority by 20 if the object fails
`DLS1(config-if)#exit`	Moves to global configuration mode

IPv4 Configuration Example: GLBP

Figure 13-3 shows the network topology for the configuration that follows, which shows how to configure GLBP using commands covered in this chapter. Note that only the commands specific to GLBP are shown in this example.

NOTE: The Gateway Load Balancing Protocol (GLBP) is not supported on the Catalyst 3750-E, 3750, 3560, or 3550 platforms. GLBP is supported on the Catalyst 4500 and Catalyst 6500 platforms and most recent router platforms (1800, 1900, 2800, 2900, 3800, 3900).

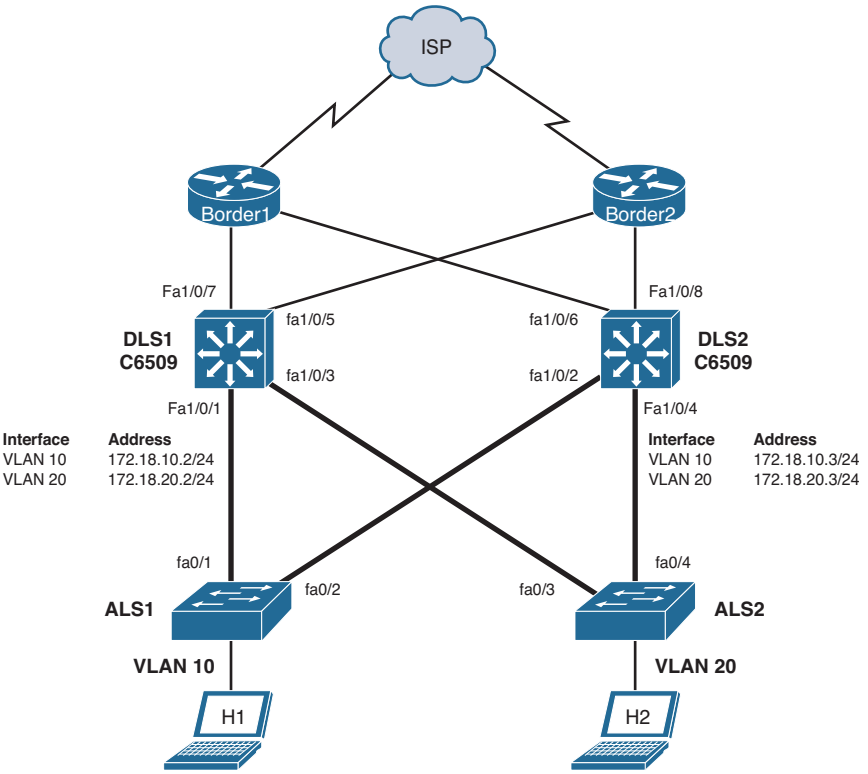


Figure 13-3 Network Topology for GLBP Configuration Example

DLS1 and DLS2 belong to GLBP groups 10 and 20. DLS1 is the AVG for GLBP group 10 and backup for GLBP group 20. DLS2 is the AVG for GLBP group 20 and backup for GLBP group 10.

DLS1 and DLS2 are responsible for the virtual IP address 172.18.10.1 on VLAN 10 and 172.18.20.1 on VLAN 20.

DLS1

`DLS1(config)#track 90 interface fastethernet1/0/7 line-protocol`	Configures tracking object 90 to monitor the line protocol on interface FastEthernet1/0/7.
`DLS1(config)#track 91 interface fastethernet1/0/5 line-protocol`	Configures tracking object 90 to monitor the line protocol on interface FastEthernet1/0/5.
`DLS1(config)#interface vlan10`	Moves to interface configuration mode.
`DLS1(config-if)#ip address 172.18.10.2 255.255.255.0`	Assigns IP address and netmask.

DLS1(config-if)#**glbp 10 ip 172.18.10.1**	Enables GLBP for group 10 on this interface with a virtual address of 172.18.10.1.
DLS1(config-if)#**glbp 10 weighting 110 lower 95 upper 105**	Specifies the initial weighting value, and the upper and lower thresholds, for a GLBP gateway. This will allow the backup AVF to start forwarding packets for VLAN 10 if an uplink fails.
DLS1(config-if)#**glbp 10 timers msec 200 msec 700**	Configures the hello timer to be 200 milliseconds and the hold timer to be 700 milliseconds.
DLS1(config-if)#**glbp 10 priority 105**	Sets the AVG priority level to 105 on the switch for VLAN 10.
DLS1(config-if)#**glbp 10 preempt delay minimum 300**	Configures the switch to take over as AVG for group 10 if this switch has a higher priority than the current active virtual gateway (AVG) after a delay of 300 seconds.
DLS1(config-if)#**glbp 10 authentication md5 keystring xyz123**	Configures the authentication key **xyz123** for GLBP packets received from the other switch in the group.
DLS1(config-if)#**glbp 10 weighting track 90 decrement 10**	Configures object 90 to be tracked in group 10. Decrement the weight by 10 if the object fails.
DLS1(config-if)#**glbp 10 weighting track 91 decrement 20**	Configures object 91 to be tracked in group 10. Decrement the weight by 20 if the object fails.
DLS1(config)#**interface vlan20**	Moves to interface configuration mode.
DLS1(config-if)#**ip address 172.18.20.2 255.255.255.0**	Assigns IP address and netmask.
DLS1(config-if)#**glbp 20 ip 172.18.20.1**	Enables GLBP for group 1 on this interface with a virtual address of 172.18.20.1.
DLS1(config-if)#**glbp 20 weighting 110 lower 95 upper 105**	Specifies the initial weighting value, and the upper and lower thresholds, for a GLBP gateway.
DLS1(config-if)#**glbp 20 timers msec 200 msec 700**	Configures the hello timer to be 200 milliseconds and the hold timer to be 700 milliseconds.
DLS1(config-if)#**glbp 20 priority 100**	Sets the AVG priority level to 100 on the switch for VLAN 20.
DLS1(config-if)#**glbp 20 preempt delay minimum 300**	Configures the switch to take over as AVG for group 10 if this switch has a higher priority than the current AVG after a delay of 300 seconds.

`DLS1(config-if)#glbp 20 authentication md5 keystring xyz123`	Configures the authentication key **xyz123** for GLBP packets received from the other switch in the group.
`DLS1(config-if)#glbp 20 weighting track 90 decrement 10`	Configures object 90 to be tracked in group 20. Decrement the weight by 10 if the object fails.
`DLS1(config-if)#glbp 20 weighting track 91 decrement 10`	Configures object 91 to be tracked in group 20. Decrement the weight by 10 if the object fails.

DLS2

`DLS2(config)#track 90 interface fastethernet1/0/8 line-protocol`	Configures tracking object 90 to monitor the line protocol on interface FastEthernet1/0/8.
`DLS2(config)#track 91 interface fastethernet1/0/6 line-protocol`	Configures tracking object 90 to monitor the line protocol on interface FastEthernet1/0/6.
`DLS2(config)#interface vlan10`	Moves to interface configuration mode.
`DLS2(config-if)#ip address 172.18.10.3 255.255.255.0`	Assigns IP address and netmask.
`DLS2(config-if)#glbp 10 ip 172.18.10.1`	Enables GLBP for group 10 on this interface with a virtual address of 172.18.10.1.
`DLS2(config-if)#glbp 10 weighting 110 lower 95 upper 105`	Specifies the initial weighting value, and the upper and lower thresholds, for a GLBP gateway.
`DLS2(config-if)#glbp 10 timers msec 200 msec 700`	Configures the hello timer to be 200 milliseconds and the hold timer to be 700 milliseconds.
`DLS2(config-if)#glbp 10 priority 100`	Sets AVG the priority level to 100 on the switch for VLAN 10.
`DLS2(config-if)#glbp 10 preempt delay minimum 300`	Configures the switch to take over as AVG for group 10 if this switch has a higher priority than the current AVG after a delay of 300 seconds.
`DLS2(config-if)#glbp 10 authentication md5 keystring xyz123`	Configures the authentication key **xyz123** for GLBP packets received from the other switch in the group.
`DLS2(config-if)#glbp 10 weighting track 90 decrement 10`	Configures object 90 to be tracked in group 10. Decrement the weight by 10 if the object fails.

`DLS2(config-if)#glbp 10 weighting track 91 decrement 20`	Configures object 91 to be tracked in group 10. Decrement the weight by 20 if the object fails.
`DLS2(config)#interface vlan20`	Moves to interface configuration mode.
`DLS2(config-if)#ip address 172.18.20.3 255.255.255.0`	Assigns IP address and netmask.
`DLS2(config-if)#glbp 20 ip 172.18.20.1`	Enables GLBP for group 1 on this interface with a virtual address of 172.18.20.1.
`DLS2(config-if)#glbp 20 weighting 110 lower 95 upper 105`	Specifies the initial weighting value, and the upper and lower thresholds, for a GLBP gateway.
`DLS2(config-if)#glbp 20 timers msec 200 msec 700`	Configures the hello timer to be 200 milliseconds and the hold timer to be 700 milliseconds.
`DLS2(config-if)#glbp 20 priority 105`	Sets the AVG priority level to 105 on the switch for VLAN 20.
`DLS2(config-if)#glbp 20 preempt delay minimum 300`	Configures the switch to take over as AVG for group 10 if this switch has a higher priority than the current AVG after a delay of 300 seconds.
`DLS2(config-if)#glbp 20 authentication md5 keystring xyz123`	Configures the authentication key **xyz123** for GLBP packets received from the other switch in the group.
`DLS2(config-if)#glbp 20 weighting track 90 decrement 10`	Configures object 90 to be tracked in group 20. Decrement the weight by 10 if the object fails.
`DLS2(config-if)#glbp 20 weighting track 91 decrement 10`	Configures object 91 to be tracked in group 20. Decrement the weight by 10 if the object fails.

IPv4 Configuration Example: VRRP on Router and L3 Switch

Figure 13-4 shows the network topology for the configuration that follows, which shows how to configure VRRP using the commands covered in this chapter. Note that only the commands specific to VRRP are shown in this example. Full routing and connectivity are assumed. R1 and DLS-2 are the participating devices in VRRP.

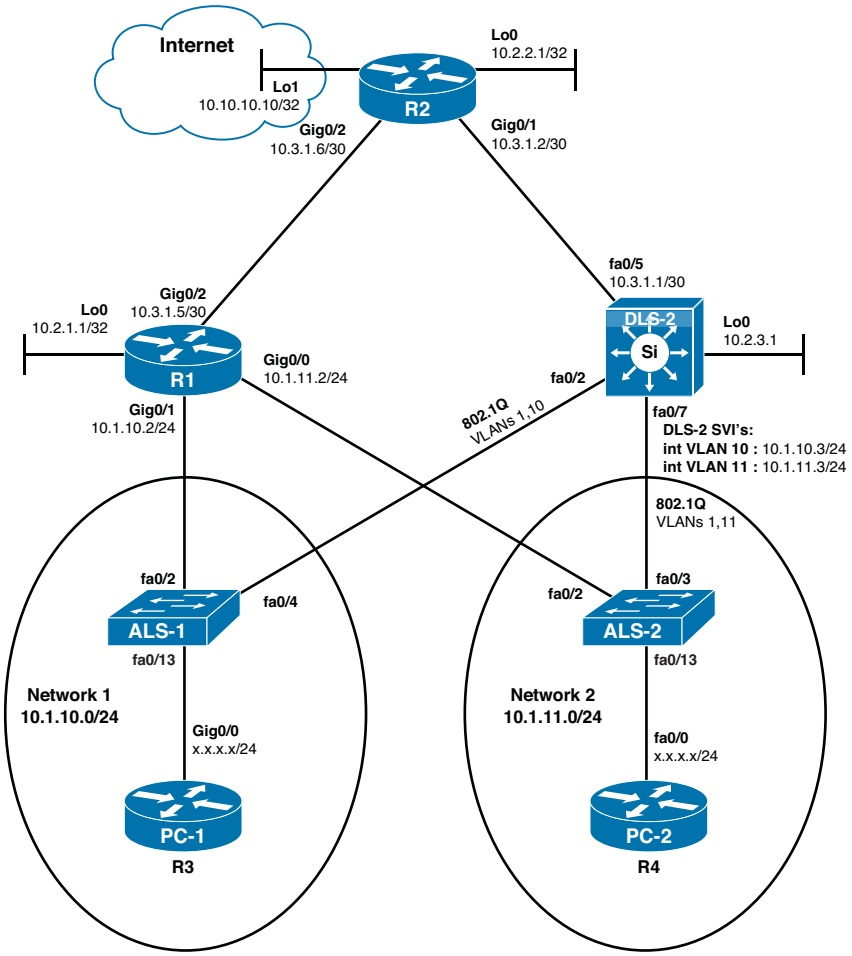


Figure 13-4 VRRP for IPv4 Using Router and L3 Switch

The network devices are configured as follows:

- R1 and DLS-2 are VRRP partners.
- ALS-1 and ALS-2 are Layer 2 switches where ALS-1 is the network switch for 10.1.10.0/24 and ALS-2 for 10.1.11.0/24.
- R1, R2, and DLS-2 are OSPF neighbors; Fast Ethernet 0/5 on DLS-2 is a routed port.
- VLAN 10 is configured on ALS-1; VLAN 11 is configured on ALS-2; DLS-2 has both VLAN 10 and 11 configured.
- All lines connecting DLS-2, ALS-1, and ALS-2 are 802.1Q trunks.
- R1 is the preferred forwarded for network 10.1.10.0/24 and DLS-2 is the preferred forwarded for network 10.1.11.0/24.

R1

`R1(config)#ip sla 10`	Enters SLA programming mode.
`R1(config-ip-sla)#icmp-echo 10.10.10.10`	Has the SLA ping 10.10.10.10.
`R1(config-ip-sla-echo)#frequency 5`	Pings 10.10.10.10 every 5 seconds.
`R1(config-ip-sla-echo)#exit`	Exits SLA programming mode.
`R1(config)#ip sla schedule 10 life forever start-time now`	Specifies the SLA start time and duration.
`R1(config)#track 100 ip sla 10`	Creates tracking object 100 calling SLA 10.
`R1(config)#track 2 interface gigabitethernet0/2 line-protocol`	Creates tracking object 2 to monitor line protocol up/down status of interface GigabitEthernet0/2.
`R1(config-track)#exit`	Exits tracking configuration mode.
`R1(config)#interface gigabitethernet0/0`	Enters interface programming mode for GigabitEthernet0/0.
`R1(config-if)#ip address 10.1.11.2 255.255.255.0`	Assigns the physical interface address of 10.1.11.2/24.
`R1(config-if)#vrrp 11 ip 10.1.11.1`	Assigns the VRRP virtual IP address of 10.1.11.1 for VRRP group 11.
`R1(config-if)#vrrp 11 authentication text CISCO123`	Use the string **CISCO123** for authentication between group 11 members.
	NOTE: Authentication by key chain is not available on some L3 switch platforms.
`R1(config-if)#vrrp 11 track 2`	Has VRRP group 11 watch tracking object 2, line protocol up/down on interface GigabitEthernet0/2.
`R1(config-if)#interface gigabitethernet0/1`	Enters interface programming mode.
`R1(config-if)#ip address 10.1.10.2 255.255.255.0`	Assigns the physical interface address of 10.1.10.2/24.
`R1(config-if)#vrrp 10 ip 10.1.10.1`	Assigns the VRRP virtual IP address of 10.1.10.1 for VRRP group 10.
`R1(config-if)#vrrp 10 priority 105`	Assigns group 10 virtual forwarder priority of 105. The default is 100.
`R1(config-if)#vrrp 10 track 2`	Has VRRP group 10 watch tracking object 2, line protocol up/down on interface GigabitEthernet0/2.

`R1(config-if)#vrrp 10 track 100 decrement 6`	Has VRRP group 10 watch a second tracking object. Object 100 looks for ICMP ping connectivity to 10.10.10.10 every 5 seconds.
`R1(config-if)#end`	Return to privileged EXEC mode.

DLS-2

`DLS-2(config)#ip sla 10`	Enters SLA 10 programming mode.
`DLS-2(config-ip-sla)#icmp-echo 10.10.10.10`	Has the SLA ping 10.10.10.10.
`DLS-2(config-ip-sla-echo)#frequency 5`	Pings 10.10.10.10 every 5 seconds.
`DLS-2(config-ip-sla-echo)#exit`	Exits SLA programming mode.
`DLS-2(config)#ip sla schedule 10 life forever start-time now`	Specifies SLA 10 start time and duration.
`DLS-2(config)#track 100 ip sla 10`	Creates tracking object 100, which calls SLA 10.
`DLS-2(config)#track 2 interface fastethernet0/5 line-protocol`	Creates tracking object 2 to monitor line protocol up/down status of interface FastEthernet0/5 (routed port to R2).
`DLS-2(config-if)#interface fastethernet0/5`	Enters interface programming mode.
`DLS-2(config-if)#no switchport`	Change FastEthernet0/5 to a Layer 3 port.
`DLS-2(config-if)#ip address 10.3.1.1 255.255.255.252`	Assigns IPv4 address 10.3.1.1/30.
`DLS-2(config)#interface fastethernet0/2`	Enters interface programming mode.
`DLS-2(config-if)#switchport trunk encapsulation dot1q`	Creates a trunk specifying 802.1Q tagging.
`DLS-2(config-if)#switchport mode trunk`	Forces trunk mode.
`DLS-2(config-if)#switchport trunk allowed vlan 1,10`	Limits VLAN traffic on this trunk to VLANs 1 and 10.
`DLS-2(config-if)#interface fastethernet0/7`	Enters interface programming mode.
`DLS-2(config-if)#switchport trunk encapsulation dot1q`	Creates a trunk specifying 802.1Q tagging.

`DLS-2(config-if)#switchport mode trunk`	Forces trunk mode.
`DLS-2(config-if)#switchport trunk allowed vlan 1,11`	Limits VLAN traffic on this trunk to VLANs 1 and 11.
`DLS-2(config-if)#interface vlan10`	Enters switched virtual interface programming mode for VLAN 10.
`DLS-2(config-if)#ip address 10.1.10.3 255.255.255.0`	Assigns IPv4 address 10.1.10.3/24.
`DLS-2(config-if)#vrrp 10 ip 10.1.10.1`	Assigns the VRRP virtual IP address of 10.1.10.1 for VRRP group 10.
`DLS-2(config-if)#vrrp 10 track 2`	Has VRRP group 10 watch tracking object 2, line protocol up/down on interface FastEthernet0/5.
`DLS-2(config-if)#interface vlan11`	Enters switched virtual interface programming mode for VLAN 11.
`DLS-2(config-if)#ip address 10.1.11.3 255.255.255.0`	Assigns IPv4 address 10.1.11.3/24.
`DLS-2(config-if)#vrrp 11 ip 10.1.11.1`	Assigns the VRRP virtual IP address of 10.1.11.1 for VRRP group 11.
`DLS-2(config-if)#vrrp 11 priority 105`	Assigns group 11 virtual forwarder priority of 105. The default is 100.
`DLS-2(config-if)#vrrp 11 authentication text CISCO123`	Uses the string **CISCO123** for authentication between group 11 members.
`DLS-2(config-if)#vrrp 11 track 2`	Has VRRP group 11 watch tracking object 2, line protocol up/down on interface FastEthernet0/5.
`DLS-2(config-if)#vrrp 11 track 100 decrement 6`	Has VRRP group 11 watch a second tracking object. Object 100 looks for ICMP ping connectivity to 10.10.10.10 every 5 seconds.
`DLS-2(config-if)#exit`	Returns to privileged EXEC mode.

IPv6 Configuration Example: HSRP on Router and L3 Switch

Figure 13-5 shows the network topology for the IPv6 HSRPv2 configuration that follows. Router R1 and L3 switch DLS-2 are the HSRP pair.

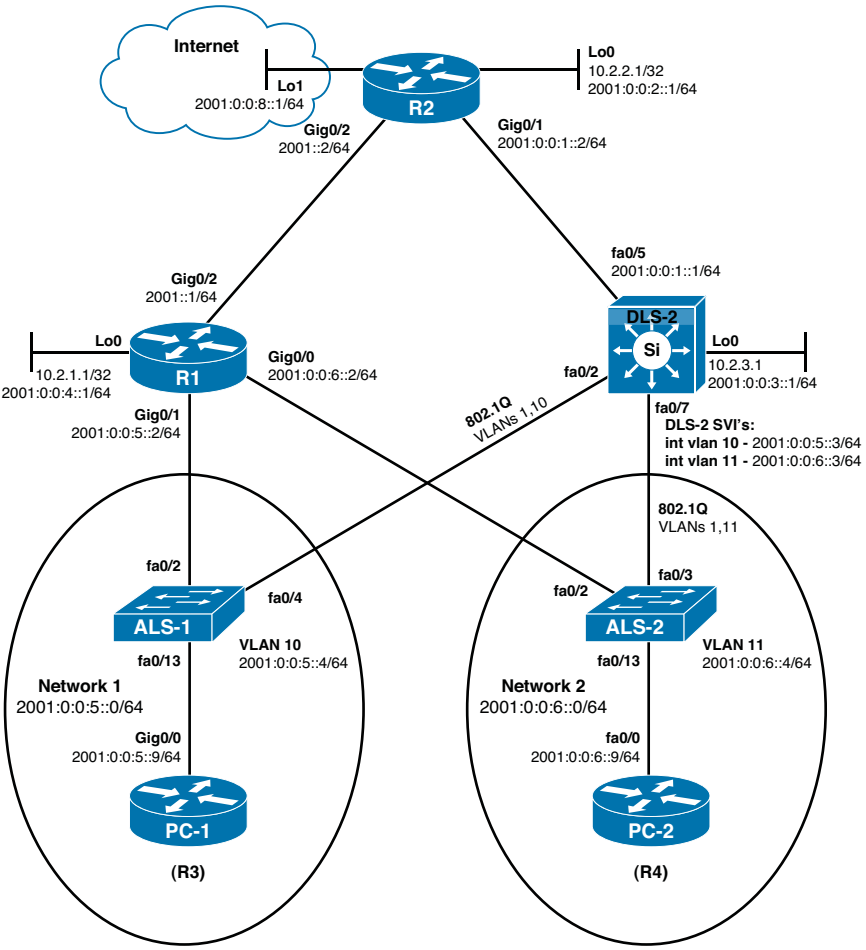


Figure 13-5 HSRPv2 IPv6 with Router and L3 Switch

R1

The network devices are configured similar to those in the previous example:

- R1 and DLS-2 are HSRPv2 partners.
- ALS-1 and ALS-2 are Layer 2 switches, where ALS-1 is the network switch for 2001:0:0:5::0/64 and ALS-2 for 2001:0:0:6::0/64.
- R1, R2, and DLS-2 are OSPFv3 neighbors; FastEthernet0/5 on DLS-2 is a routed port.
- VLAN 10 is configured on ALS-1; VLAN 11 is configured on ALS-2; DLS-2 has both VLAN 10 and 11 configured.
- All lines connecting DLS-2, ALS-1, and ALS-2 are 802.1Q trunks.
- R1 is the preferred forwarder for network 2001:0:0:5::0/64 and DLS-2 is the preferred forwarder for network 2001:0:0:6::0/64.

`R1(config)#ipv6 unicast-routing`	Enables IPv6 forwarding.
`R1(config)#ip sla 11`	Enters SLA programming mode for process 11.
`R1(config-ip-sla)#icmp-echo 2001:0:0:8::1 source-interface gigabitethernet0/2`	Has the SLA ping 2001:0:0:8::1.
`R1(config-ip-sla-echo)#frequency 5`	Pings every 5 seconds.
`R1(config-ip-sla-echo)#exit`	Exits SLA programming mode.
`1(config)#ip sla schedule 11 life forever start-time now`	Defines the start and duration for SLA 11.
`R1(config)#track 111 ip sla 11`	Creates tracking object 111 that uses SLA 11.
`R1(config-track)#exit`	Exits tracking.
`R1(config)#interface gigabitethernet0/0`	Enters interface configuration mode.
`R1(config-if)#ipv6 address 2001:0:0:6::2/64`	Assigns IPv6 unicast address.
`R1(config-if)#standby version 2`	Enables HSRP Version 2.
	NOTE: HSRP Version 2 is required for IPv6 implementation.
`R1(config-if)#standby 11 ipv6 autoconfig`	Creates IPv6 HSRP virtual address.
	NOTE: When you enter the **standby ipv6** command, a modified EUI-64 format interface identifier is generated in which the EUI-64 interface identifier is created from the relevant HSRP virtual MAC address.
	NOTE: The **standby group ipv6** interface command can offer different options when using different platforms. For example, a 3560 L3 switch will allow an IPv6 **prefix** argument, whereas a 2911G2 router will not.
`R1(config-if)#standby 11 preempt`	This device will preempt, or take control of, the active forwarding if the local priority is higher than any of the other members of the HSRP group.
	NOTE: The same preempt command arguments are available for IPv6 as in IPv4.

`R1(config-if)#standby 11 track gigabitethernet0/2 12`	Instructs HSRPv2 to follow the line protocol of GigabitEthernet0/2 and decrement the interface group priority by 12 when the interface goes down.
	NOTE: When the preceding tracking command is entered, the router creates the following line protocol tracking object: **track x interface GigabitEthernet0/2 line-protocol**, where **x** is the next available number available for a tracking object.The IOS then substitutes the tracking command **standby 11 track x decrement 12** at the interface (as seen below).
`R1(config-if)#standby 11 track 1 decrement 12`	Haves HSRP group 11 watch tracking object 1, line protocol up/down on interface GigabitEthernet0/2.
`R1(config-if)#interface gigabitethernet0/1`	Enters Interface configuration mode.
`R1(config-if)#ipv6 address 2001:0:0:5::2/64`	Assigns an IPv6 unicast address.
`R1(config-if)#standby version 2`	Select HSRP Version 2.
`R1(config-if)#standby 10 ipv6 autoconfig`	Creates IPv6 HSRP virtual address.
`R1(config-if)#standby 10 priority 105`	Sets a priority of 105 for standby group 10 on this interface.
`R1(config-if)#standby 10 preempt`	This device will preempt, or take control of, the active forwarding if the local priority is higher than any of the other members of the HSRP group.
`R1(config-if)#standby 10 track 1 decrement 12`	Links tracking object 1 to this HSRP group and decrease this device's priority by 12 when tracking object 1 is asserted.
`R1(config-if)#standby 10 track 111 decrement 7`	Links a second tracking object to this HSRP group and decrease the device's priority by 7 when asserted.

DLS-2

`DLS-2(config)#ip routing`	Enables IOS Layer 3 functionality.
`DLS-2(config)#ipv6 unicast-routing`	Enables IOS IPv6 Layer 3 functionality
`DLS-2(config)#sdm prefer dual-ipv4-and-ipv6`	Configures the Switching Database Manager on the switch to optimize memory and operating system for both IPv4 and IPv6 Layer 3 forwarding.

DLS-2(config)#**ip sla 11**	Creates and enters SLA 11.
	NOTE: The SLAs are added only as an illustration of capability.
	NOTE: There seems to be no distinction between IPv4 and IPv6 in the **ip sla** command.
DLS-2(config-ip-sla)#**icmp-echo 2001:0:0:8::1**	Assigns 2001:0:0:8::1 as the ICMP ping destination for this SLA.
DLS-2(config-ip-sla-echo)#**frequency 5**	Sends pings every 5 seconds.
DLS-2(config-ip-sla-echo)#**exit**	Exits SLA configuration mode.
DLS-2(config)#**ip sla schedule 11 life forever start-time now**	Assigns the start time and duration for SLA 11.
DLS-2(config)#**track 101 ip sla 11**	Creates tracking object 101, which uses SLA 11.
DLS-2(config-track)#**exit**	Exits tracking configuration mode.
DLS-2(config)#**interface loopback0**	Enters interface configuration mode.
DLS-2(config-if)#**ipv6 address 2001:0:0:3::1/64**	Assigns an IPv6 unicast address.
DLS-2(config-if)#**interface fastethernet0/5**	Enters interface configuration mode.
DLS-2(config-if)#**no switchport**	Changes Layer 2 switch port to a Layer 3 routed port.
DLS-2(config-if)#**ipv6 address 2001:0:0:1::1/64**	Assigns an IPv6 address to this L3 forwarding port.
DLS-2(config-if)#**interface fastethernet0/2**	Enters interface configuration mode for L2 interface.
DLS-2(config-if)#**switchport trunk encapsulation dot1q**	Enables 802.1Q trunking to ALS-1.
DLS-2(config-if)#**switchport trunk allowed vlan 1,10**	Permits traffic from VLAN 1 and 10 on the trunk.
DLS-2(config-if)#**switchport mode trunk**	Sets the port to trunk unconditionally.
DLS-2(config-if)#**interface fastethernet0/7**	Enters interface configuration mode.
DLS-2(config-if)#**switchport trunk encapsulation dot1q**	Enables 802.1Q trunking to ALS-2.
DLS-2(config-if)#**switchport trunk allowed vlan 1, 11**	Permits traffic from VLAN 1 and 11 on the trunk.
DLS-2(config-if)#**switchport mode trunk**	Sets the port to trunk unconditionally.
DLS-2(config-if)#**interface vlan10**	Enters interface programming mode for VLAN 10 SVI.

DLS-2(config-if)#**standby version 2**	Specifies HSRP Version 2.
DLS-2(config-if)#**ipv6 address 2001:0:0:5::3/64**	Assigns IPv6 unicast address.
DLS-2(config-if)#**standby 10 ipv6 autoconfig**	Creates IPv6 HSRP virtual address.
	NOTE: When you enter the **standby ipv6** command, a modified EUI-64 format interface identifier is generated in which the EUI-64 interface identifier is created from the relevant HSRP virtual MAC address.
	NOTE: The **standby group ipv6** interface command can offer different options when using different platforms. For example, a 3560 L3 switch will allow an IPv6 **prefix** argument, whereas a 2911G2 router will not.
DLS-2(config-if)#**standby 10 preempt**	Enables this group's HSRP forwarder to become active at any time when its group priority is the highest.
DLS-2(config-if)#**standby 10 track 111 decrement 10**	Links tracking object 111 to this standby group and decrease this device's priority by 10 when tracking object 111 is asserted.
DLS-2(config-if)#**interface vlan11**	Enters interface programming mode for VLAN 11 SVI.
DLS-2(config-if)#**ipv6 address 2001:0:0:6::3/64**	Assigns IPv6 unicast address.
DLS-2(config-if)#**standby version 2**	Specifies HSRP Version 2.
DLS-2(config-if)#**standby 11 ipv6 autoconfig**	Creates IPv6 HSRP virtual address.
DLS-2(config-if)#**standby 11 priority 105**	Sets a priority of 105 for standby group 11 on this interface.
DLS-2(config-if)#**standby 11 preempt**	Enables this group's HSRP forwarder to transition to active at any time when its group priority is the highest.
DLS-2(config-if)#**standby 11 track 111 decrement 10**	Link tracking object 111 to HSRP group 11 and decrease this device's priority by 10 when tracking object 111 is asserted.

NOTE: HSRP verification and **debug** commands are the same for IPv4 and IPv6.

This page intentionally left blank

CHAPTER 14

Campus Network Security

This chapter provides information about the following topics:

- Switch security recommended practices
- Configuring switch port security
  - Sticky MAC addresses
  - Verifying switch port security
- Recovering automatically from error-disabled ports
  - Verifying autorecovery of error-disabled ports
- Configuring port access lists
  - Creating and applying named MAC extended ACLs
- Configuring storm control
- Implementing authentication methods
  - Local database authentication
  - RADIUS authentication
    - Legacy configuration for RADIUS servers
    - Modular configuration for RADIUS server
  - TACACS+ authentication
    - Legacy configuration for TACACS+ servers
    - Modular configuration for TACACS+ servers
  - Configuring authorization and accounting
    - Authorization
    - Accounting
  - Configuring 802.1x port-based authentication
- Configuring DHCP snooping
  - Verifying DHCP Snooping
- IP Source Guard
- Dynamic ARP Inspection (DAI)
  - Verifying DAI
- Mitigating VLAN hopping: best practices

- VLAN access lists
  - Verifying VACLs
  - Configuration example: VACLs
- Private VLANs
  - Verifying PVLANs
  - Configuration example: PVLANs

CAUTION: Your hardware platform or software release might not support all the commands documented in this chapter. Please refer to the Cisco website for specific platform and software release notes.

Switch Security Recommended Practices

Layer 2 security implementation is often forgotten. However, you should take the basic security measures to guard against a host of attacks that can be launched at a switch and its ports. Here are some of the recommended best practices for switch security.

Table 14-1 shows the checklist that should be used when securing a Cisco Catalyst switch.

TABLE 14-1 Switch Security Recommended Practices

Recommended Practices	Y/N
Configure secure passwords (**enable secret**)	
Use encrypted passwords (**service password-encryption**)	
Use external AAA authentication	
Use system banners (**banner motd** and **banner login**)	
Secure console and vty access using passwords and access control lists (ACLs)	
Secure web interface (**no ip http server / no ip http secure-server**) or with ACLs	
Use Secure Shell (SSH) instead of Telnet	
Secure SNMP access (disable “write” community)	
Secure STP operation (BPDU Guard)	
Disable Cisco Discovery Protocol (CDP) when not required	
Secure unused switch ports (use **shutdown** command, configure static access mode, and place port in unused VLAN)	

Configuring Switch Port Security

`Switch(config)#interface fastethernet0/1`	Moves to interface configuration mode.
`Switch(config-if)#switchport mode access`	A required step, this sets the interface to access mode (as opposed to trunk mode).
	NOTE: A port cannot be secured while in the default dynamic auto mode.
`Switch(config-if)#switchport port-security`	Enables port security on the interface.
`Switch(config-if)#switchport port-security maximum 4`	Sets a maximum limit of 4 MAC addresses that will be allowed on this port (default maximum is 1).
	NOTE: The maximum number of secure MAC addresses that you can configure on a switch is set by the maximum number of available MAC addresses allowed in the system.
`Switch(config-if)#switchport port-security mac-address 1234.5678.90ab`	Sets a specific secure MAC address 1234.5678.90ab. You can add additional secure MAC addresses up to the maximum value configured.
`Switch(config-if)#switchport port-security violation shutdown`	Configures port security to shut down the interface if a security violation occurs.
	NOTE: In shutdown mode, the port is err-disabled, a log entry is made, and manual intervention or err-disable recovery must be used to reenable the interface.
`Switch(config-if)#switchport port-security violation restrict`	Configures port security to restrict mode if a security violation occurs.
	NOTE: In restrict mode, frames from a nonallowed address are dropped, and a log entry is made. The interface remains operational.
`Switch(config-if)#switchport port-security violation protect`	Configures port security to protect mode if a security violation occurs.
	NOTE: In protect mode, frames from a nonallowed address are dropped, but no log entry is made. The interface remains operational.

Sticky MAC Addresses

Sticky MAC addresses are a feature of port security. Sticky MAC addresses limit switchport access to a specific MAC address that can be dynamically learned, as opposed to a network administrator manually associating a MAC addresses with a specific switch

port. These addresses are stored in the running configuration file. If this file is saved, the sticky MAC addresses will not have to be relearned when the switch is rebooted, providing a high level of switch port security.

`Switch(config)#interface fastethernet0/5`	Moves to interface configuration mode.
`Switch(config-if)#switchport port-security mac-address sticky`	Converts all dynamic port security-learned MAC addresses to sticky secure MAC addresses.
`Switch(config-if)#switchport port-security mac-address sticky vlan 10 voice`	Converts all dynamic port security-learned MAC addresses to sticky secure MAC addresses on voice VLAN 10.
	NOTE: The **voice** keyword is available only if a voice VLAN is first configured on a port and if that VLAN is not the access VLAN.

Verifying Switch Port Security

`Switch#show port-security`	Displays security information for all interfaces.
`Switch#show port-security interface fastethernet0/5`	Displays security information for interface FastEthernet0/5.
`Switch#show port-security address`	Displays all secure MAC addresses configured on all switch interfaces.
`Switch#show mac address-table [dynamic]`	Displays the entire MAC address table or simply the dynamic addresses learned.
`Switch#clear mac address-table dynamic`	Deletes all dynamic MAC addresses.
`Switch#clear mac address-table dynamic address aaaa.bbbb.cccc`	Deletes the specified dynamic MAC address.
`Switch#clear mac address-table dynamic interface fastethernet0/5`	Deletes all dynamic MAC addresses on interface FastEthernet0/5.
`Switch#clear mac address-table dynamic vlan 10`	Deletes all dynamic MAC addresses on VLAN 10.
`Switch#clear mac address-table notification`	Clears MAC notification global counters.
	NOTE: Beginning with Cisco IOS Software Release 12.1(11)EA1, the **clear mac address-table** command (no hyphen) replaces the **clear mac-address-table** command (with the hyphen). The **clear mac-address-table static** command (with the hyphen) will become obsolete in a future release.

Recovering Automatically from Error-Disabled Ports

You can also configure a switch to autorecover error-disabled ports after a specified amount of time. By default, the autorecovery feature is disabled.

`Switch(config)#errdisable recovery cause psecure-violation`	Enables the timer to recover from a port security violation disable state.
`Switch(config)#errdisable recovery interval` *seconds*	Specifies the time to recover from the error-disabled state. The range is 30 to 86,400 seconds. The default is 300 seconds.
	TIP: Disconnect the offending host; otherwise, the port will remain disabled, and the violation counter will be incremented.

Verifying Autorecovery of Error-Disabled Ports

`Switch#show errdisable recovery`	Displays error-disabled recovery timer information associated with each possible reason the switch could error disable a port
`Switch#show interfaces status err-disabled`	Displays interface status or a list of interfaces in error-disabled state
`Switch#clear errdisable interface` *interface-id* `vlan [`*vlan-list*`]`	Reenables all or specified VLANs that were error-disabled on an interface

Configuring Port Access Lists

Port ACLs (PACLs) are ACLs that are applied to Layer 2 interfaces on a switch. PACLs are supported only on physical interfaces and not on EtherChannel interfaces. PACLs can be applied on outbound and inbound interfaces. The following access lists are supported:

- Standard IP access lists using source addresses
- Extended IP access lists using source and destination addresses and optional protocol type information
- MAC extended access lists using source and destination MAC addresses and optional protocol type information

The switch examines ACLs on an interface and permits or denies packet forwarding based on how the packet matches the entries in the ACL. In this way, ACLs control access to a network or to part of a network. Port ACLs are applied only on the ingress traffic. The PACL feature does not affect Layer 2 control packets, such as CDP, VTP, DTP, and STP, received on the port.

Creating and Applying Named Port Access List

`Switch(config)#mac access-list extended MAC-FILTER`	Defines an extended MAC access list using the name MAC-FILTER.
`Switch(config-ext-macl)#permit host aabb.ccdd.eeff any`	Permits the device with a MAC address of **aabb.ccdd.eeff**.
	TIP: As with IP ACLs, an implicit **deny any any** is assumed at the end of all MAC PACLs.
`Switch(config-ext-macl)#exit`	Returns to global configuration mode.
`Switch(config)#interface gigabitethernet0/1`	Identifies a specific interface, and enters interface configuration mode. The interface must be a physical Layer 2 interface.
`Switch(config-if)#mac access-group MAC-FILTER in`	Controls access to the specified interface by using the MAC-FILTER access list.

NOTE: For further information on port access lists, see the "Catalyst 2960-X Switch Security Configuration Guide, Cisco IOS Release 15.0(2)EX" on Cisco.com: http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst2960x/software/15-0_2_EX/security/configuration_guide/b_sec_152ex_2960-x_cg.html.

Configuring Storm Control

The storm control feature prevents LAN ports from being disrupted by a broadcast, multicast, or unicast traffic storm on physical interfaces and is used to protect against or isolate broadcast storms caused by STP misconfigurations, unicast storms created by malfunctioning hosts, or denial-of-service (DoS) attacks. Storm control configuration is done per interface for each type of traffic separately. Storm control is typically configured on access ports, to limit the effect of traffic storm on access level, before it enters the network.

`Switch(config)#interface gigabitethernet0/1`	Moves to interface configuration mode
`Switch(config-if)#storm-control broadcast level 75.5`	Enables broadcast storm control with a 75.5 percent rising suppression level
`Switch(config-if)#storm-control unicast level bps 50m`	Enables unicast storm control on a port with a 50 million bits per second rising suppression level
`Switch(config-if)#storm-control multicast level pps 2k 1k`	Enables multicast storm control on a port with a 2000-packets-per-second rising suppression level and a 1000-packets-per-second falling suppression level
`Switch(config-if)#storm-control action shutdown`	Disables the port during a storm
`Switch(config-if)#storm-control action trap`	Sends an SNMP trap when a storm occurs

NOTE: Use the **show storm-control** command in EXEC mode to display broadcast, multicast, or unicast storm control settings on the switch or on the specified interface or to display storm-control history.

Implementing Authentication Methods

Authentication, authorization, and accounting (AAA) is a standards-based framework that you can implement to control who is permitted to access a network (authenticate), what they can do while they are there (authorize), and audit what actions they performed while accessing the network (accounting).

Local Database Authentication

`Switch(config)#username ADMIN privilege 15 secret cisco123`	Creates an entry in the local database with a privilege level of 15 and a message digest 5 (MD5) authentication encrypted password
`Switch(config)#aaa new-model`	Enables AAA access control mode
`Switch(config)#aaa authentication login default local-case enable`	Defines the default authentication method list to authenticate to the case-sensitive local database first and the enable password second
`Switch(config)#aaa authentication login vty local line`	Defines the authentication method list vty to authenticate to the local database first and the vty line password second
`Switch(config)#line vty 0 15`	Enters the vty configuration mode
`Switch(config-line)#login authentication vty`	Specifies the AAA service to use the authentication method list vty when a user logs in
`Switch(config-line)#exit`	Returns to global configuration mode
`Switch(config)#line console 0`	Enters console 0 configuration mode
`Switch(config-line)#login authentication default`	Specifies the AAA service to use the default method list when a user logs in

NOTE: A method list describes the sequence and authentication methods to be queried to authenticate a user. The software uses the first method listed to authenticate users; if that method fails to respond, the software selects the next authentication method in the method list. This process continues until there is successful communication with a listed authentication method or until all defined methods are exhausted. If authentication fails at any point in this cycle, the authentication process stops, and no other authentication methods are attempted.

RADIUS Authentication

RADIUS is fully open standard protocol (RFC 2865 and 2866). RADIUS uses UDP port 1812 for the authentication and authorization, and port 1813 for accounting (or ports 1645 and 1646 if using the default Cisco values).

Legacy Configuration for RADIUS Servers

The traditional approach to configure a RADIUS server on a Cisco IOS device would be with the **radius-server** global configuration command.

`Switch(config)#username admin secret cisco`	Creates user with username admin and encrypted password cisco.
`Switch(config)#aaa new-model`	Enables AAA access control mode.
`Switch(config)#radius-server host 192.168.55.12 auth-port 1812 acct-port 1813 key S3CR3TKEY`	Specifies a RADIUS server at 192.168.55.12 with S3CR3TKEY as the authentication key using UDP port 1812 for authentication requests and UDP port 1813 for accounting requests.
`Switch(config)#aaa authentication login default group radius local line`	Sets login authentication for the default method list to authenticate to the RADIUS server first, locally defined users second, and use the line password as the last resort.
`Switch(config)#aaa authentication login NO_AUTH none`	Specifies the authentication method list NO_AUTH to require no authentication.
`Switch(config)#line vty 0 15`	Moves to vty configuration mode.
`Switch(config-line)#login authentication default`	Specifies the AAA service to use the default method list when a user logs in via vty.
`Switch(config-line)#password S3cr3Tw0Rd`	Specifies a vty line password on lines 0 through 15.
`Switch(config-line)#line console 0`	Moves to console 0 configuration mode.
`Switch(config-line)#login authentication NO_AUTH`	Specifies the AAA service to use the authentication method list NO_AUTH when a user logs in via the console port.
	NOTE: If authentication is not specifically set for a line, the default is to deny access and no authentication is performed.

Modular Configuration for RADIUS Server

The legacy configuration method outlined earlier in this chapter will soon be deprecated. The new approach, which is not supported across all platforms or IOS versions yet, brings modularity and consistency when configuring RADIUS in both IPv4 and IPv6 environments. The new method is configured in three steps: One sets the RADIUS

server parameters, one defines the RADIUS server group, and one defines the AAA commands to use RADIUS.

`Switch(config)#aaa new-model`	Enables AAA access control mode.
`Switch(config)#radius server RADSRV`	Specifies the name **RADSRV** for the RADIUS server configuration and enters RADIUS server configuration mode.
`Switch(config-radius-server)#address ipv4 192.168.100.100 auth-port 1645 acct-port 1646`	Configures the IPv4 address for the RADIUS server, as well as the accounting and authentication parameters.
`Switch(config-radius-server)#key C1sc0`	The shared secret key that is configured on the RADIUS server must be defined for secure RADIUS communications.
`Switch(config-radius-server)#exit`	Returns to global configuration mode
`Switch(config)#ip radius source-interface vlan900`	To force RADIUS to use the IP address of a specified interface for all outgoing RADIUS packets. This is a global configuration command.
`Switch(config)#aaa group server radius RADSRVGRP`	Defines a RADIUS server group called **RADSRVGRP**.
`Switch(config-sg-radius)#server name RADSRV`	Adds the RADIUS server **RADSRV** to the **RADSRVGRP** group.
`Switch(config-sg-radius)#exit`	Returns to global configuration mode
`Switch(config)#aaa authentication login RAD_LIST group RADSRVGRP local`	Configures login authentication using a method list called **RAD_LIST,** which uses **RADSRVGRP** as the primary authentication option and local user database as a backup.
`Switch(config)#line vty 0 4`	Moves to VTY configuration mode
`Switch(config)#login authentication RAD_LIST`	Applies the **RAD_LIST** method list to the VTY lines.

TACACS+ Authentication

TACACS+ is Cisco proprietary protocol not compatible with the older versions such as TACACS or XTACACS, which are now deprecated. It allows for greater modularity, by total separation of all three AAA functions. TACACS+ uses TCP port 49, and thus reliability is ensured by the transport protocol itself. Entire TACACS+ packet is encrypted, so communication between NAS and the TACACS+ server is completely secure.

Legacy Configuration for TACACS+ Servers

The traditional approach to configure a TACACS+ server on a Cisco IOS device would be with the **tacacs-server** global configuration command.

`Switch(config)#username admin secret cisco`	Creates user with username admin and encrypted password cisco.
`Switch(config)#aaa new-model`	Enables AAA access control mode.
`Switch(config)#tacacs-server host 192.168.55.13 single-connection key C1sc0`	Specifies a TACSCS+ server at 192.168.55.13 with an encryption key of C1sc0. The single-connection keyword maintains a single open connection between the switch and the server.
`Switch(config)#aaa authentication login TACSRV group tacacs+ local`	Sets login authentication for the **TACSRV** method list to authenticate to the TACACS+ server first, and the locally defined username and password second.
`Switch(config)#line console 0`	Moves to console 0 configuration mode.
`Switch(config-line)#login authentication TACSRV`	Specifies the AAA service to use the **TACSRV** authentication method list when users connect to the console port.

Modular Configuration for TACACS+ Servers

Similar to the RADIUS modular configuration shown in the previous section, it is possible to use a modular approach when configuring TACACS+. The same three steps apply (define TACACS+ server parameters, define TACACS+ server group, define AAA commands).

`Switch(config)#aaa new-model`	Enables AAA access control mode
`Switch(config)#tacacs server TACSRV`	Specifies the name **TACSRV** for the TACACS+ server configuration and enters TACACS+ server configuration mode
`Switch(config-server-tacacs)#address ipv4 192.168.100.200`	Configures the IPv4 address for the TACACS+ server
`Switch(config- server-tacacs)#key C1sc0`	The shared secret key that's configured on the TACACS+ server must be defined for secure TACACS+ communications
`Switch(config- server-tacacs)#single-connection`	Enables all TACACS+ packets to be sent to the same server using a single TCP connection
`Switch(config-server-tacacs)#exit`	Returns to global configuration mode
`Switch(config)#aaa group server tacacs+ TACSRVGRP`	Defines a TACACS+ server group called **TACSRVGRP**

`Switch(config-sg-tacacs+)#server name TACSRV`	Adds the TAACACS+ server **TACSRV** to the **TACSRVGRP** group
`Switch(config-sg-tacacs+)#exit`	Returns to global configuration mode
`Switch(config)#aaa authentication login TAC_LIST group TACSRVGRP local`	Configures login authentication using a method list called **TAC_LIST**, which uses **TACSRVGRP** as the primary authentication option and local user database as a backup
`Switch(config)#line vty 0 4`	Moves to VTY configuration mode
`Switch(config-line)#login authentication TAC_LIST`	Applies the **TAC_LIST** method list to the VTY lines.

Configuring Authorization and Accounting

After AAA has been enabled on a Cisco IOS device and **aaa authentication** has been configured, you can optionally configure **aaa authorization** and **aaa accounting**.

Authorization

Configuring authorization is a two-step process. First a method list is defined, and then it is applied to a corresponding interface or line.

`Switch(config)#aaa authorization exec default group radius group tacacs+ local`	Defines the default EXEC authorization method list, which uses the RADIUS servers first, the TACACS+ servers second, and the local user database as backup
`Switch(config-line)#line vty 0 15`	Moves to vty configuration mode
`Switch(config-if)#authorization exec default`	Applies the default authorization list to the vty lines

Accounting

Configuring accounting is also a two-step process. First a method list is defined, and then it is applied to a corresponding interface or line.

`Switch(config)#aaa accounting exec default start-stop group radius`	Defines the default EXEC accounting method list to send, to the RADIUS server, a start accounting notice at the beginning of the requested event and a stop accounting notice at the end of the event.
`Switch(config)#line vty 0 15`	Moves to vty configuration mode.
`Switch(config-line)#accounting exec default`	Applies the default accounting list to the vty lines

Configuring 802.1x Port-Based Authentication

The IEEE 802.1x standard defines a client/server-based access control and authentication protocol that prevents unauthorized clients from connecting to a LAN through switch ports unless they are properly authenticated. The authentication server authenticates each client connected to a switch port before any services offered by the switch or the LAN behind it are made available.

`Switch(config)#aaa new-model`	Enables AAA.
`Switch(config)#radius-server host 192.168.55.12 auth-port 1812 key S3CR3TKEY`	Specifies a RADIUS server at 192.168.55.12 with S3CR3TKEY as the authentication key using UDP port 1812 for authentication requests.
`Switch(config)#aaa authentication dot1x default group radius`	Creates an 802.1x port-based authentication method list. This method specifies using a RADIUS server for authentication.
	NOTE: When using the **aaa authentication dot1x** command, you must use at least one of the following keywords: **group radius:** Use a list of RADIUS servers for authentication. **none:** Use no authentication. The client is automatically authenticated without the switch using information supplied by the client. This method should only be used as a second method. If the first method of **group radius** is not successful, the switch will use the second method for authentication until a method is successful. In this case, no authentication would be used.
`Switch(config)#dot1x system-auth-control`	Globally enables 802.1x port-based authentication.
`Switch(config)#interface fastethernet0/1`	Moves to interface configuration mode.
`Switch(config-if)#dot1x port-control auto`	Enables 802.1x authentication on this interface.
	NOTE: The **auto** keyword allows the port to begin in the unauthorized state. This will allow only Extensible Authentication Protocol over LAN (EAPOL) frames to be sent and received through the port. Other keywords available here include the following: **force-authorized:** Disables 802.1x authentication and causes the port to transition to the authorized state without any authentication exchange required. This is the default setting. **force-unauthorized:** Causes the port to remain in the unauthorized state, ignoring all attempts by the client to authenticate. The switch cannot provide authentication services to the client through the interface.

	NOTE: You will not be able to issue **dot1x** commands on the interface if it is not set to **switchport mode access**.
`Switch#`**`show dot1x`**	Verifies your 802.1x entries.

Configuring DHCP Snooping

Dynamic Host Configuration Protocol (DHCP) snooping is a DHCP security feature that provides network security by filtering untrusted DHCP messages and by building and maintaining a DHCP snooping binding database, which is also referred to as a DHCP snooping binding table.

`Switch(config)#`**`ip dhcp snooping`**	Enables DHCP snooping globally.
	NOTE: If you enable DHCP snooping on a switch, the following DHCP relay agent commands are not available until snooping is disabled: Switch(config)**#ip dhcp relay information check** Switch(config)**#ip dhcp relay information policy** {**drop** | **keep** | **replace**} Switch(config)**#ip dhcp relay information trust-all** Switch(config-if)**#ip dhcp relay information trusted** If you enter these commands with DHCP snooping enabled, the switch returns an error message.
`Switch(config)#`**`ip dhcp snooping vlan 20`**	Enables DHCP snooping on VLAN 20.
`Switch(config)#`**`ip dhcp snooping vlan 10-35`**	Enables DHCP snooping on VLANs 10–35.
`Switch(config)#`**`ip dhcp snooping vlan 20 30`**	Enables DHCP snooping on VLANs 20–30.
`Switch(config)#`**`ip dhcp snooping vlan 10,12,14`**	Enables DHCP snooping on VLANs 10, 12, and 14.
`Switch(config)#`**`ip dhcp snooping information option`**	Enables DHCP option 82 insertion.
	NOTE: DHCP address allocation is usually based on an IP address, either the gateway IP address or the incoming interface IP address. In some networks, you might need additional information to determine which IP address to allocate. By using the "relay agent information option" (option 82), the Cisco IOS relay agent can include additional information about itself when forwarding DHCP packets to a DHCP server. The relay agent will add the circuit identifier suboption and the remote ID suboption to the relay information option and forward this all to the DHCP server.

`Switch(config)#interface fastethernet0/1`	Moves to interface configuration mode.
`Switch(config-if)# switchport trunk encapsulation dot1q`	Creates an uplink trunk with 802.1q encapsulation.
`Switch(config-if)# switchport mode trunk`	Force the switch port to be a trunk.
`Switch(config-if)# switchport trunk allowed vlan 10,20`	Select VLANs that are allowed transport on the trunk.
`Switch(config-if)#ip dhcp snooping trust`	Configures the interface as trusted.
	NOTE: There must be at least one trusted interface when working with DHCP snooping. It is usually the port connected to the DHCP server or to uplink ports.By default, all ports are untrusted.
`Switch(config-if)#ip dhcp snooping limit rate 75`	Configures the number of DHCP packets per second that an interface can receive.
	NOTE: The range of packets that can be received per second is 1 to 4,294,967,294. The default is no rate configured.
	TIP: Cisco recommends an untrusted rate limit of no more than 100 packets per second.
`Switch(config-if)#ip dhcp snooping verify mac-address`	Configures the switch to verify that the source MAC address in a DHCP packet that is received on an untrusted port matches the client hardware address in the packet.

Verifying DHCP Snooping

`Switch#show ip dhcp snooping`	Displays the DHCP snooping configuration for a switch
`Switch#show ip dhcp snooping binding`	Displays only the dynamically configured bindings in the DHCP snooping binding database
`Switch#show ip source binding`	Display the dynamically and statically configured bindings
`Switch#show running-config`	Displays the status of the insertion and removal of the DHCP option 82 field on all interfaces

IP Source Guard

IP Source Guard prevents a malicious host from hijacking its neighbor's IP address. IP Source Guard dynamically maintains a per-port table with IP-to-MAC-to-switch port

bindings. This is usually accomplished with the accumulated DHCP snooping data. The binding table can also be manually populated.

`Switch(config)#ip dhcp snooping`	Enables DHCP snooping, globally.
`Switch(config)#ip dhcp snooping vlan 10-35`	Enables DHCP snooping on VLANs 10–35.
`Switch(config)#interface fastethernet0/1`	Moves to interface configuration mode.
`Switch(config-if)#ip verify source`	Enables IP Source Guard with IP address filtering on the port.
`Switch(config-if)#ip verify source port-security`	Enables IP Source Guard with IP and MAC address filtering on the port.
`Switch(config)#exit`	Exits interface configuration mode.
`Switch(config)#ip source binding 0000.1111.2222 vlan 35 10.1.1.1 interface gigabitethernet1/0/1`	Add a static IP source binding between MAC 0000.1111.2222, VLAN 35, address 10.1.1.1, and interface GigabitEthernet1/0/1
`Switch#show ip source binding`	Display the IP source bindings on a switch.
`Switch#show ip verify source`	Display the IP source guard configuration on the switch or on a specific interface.
	NOTE: IP Source Guard is not supported on EtherChannels.

Dynamic ARP Inspection

Dynamic ARP Inspection (DAI) determines the validity of an ARP packet. This feature prevents attacks on the switch by not relaying invalid ARP requests and responses to other ports in the same VLAN.

`Switch(config)#ip dhcp snooping`	Enables DHCP snooping, globally.
`Switch(config)#ip dhcp snooping vlan 10-20`	Enables DHCP snooping on VLANs 10–20.
`Switch(config)#ip arp inspection vlan 10-20`	Enables DAI on VLANs 10 to 20, inclusive.
`Switch(config)#ip arp inspection validate src-mac`	Configures DAI to drop ARP packets when the source MAC address in the body of the ARP packet does not match the source MAC address specified in the Ethernet header. This check is performed on both ARP requests and responses.

`Switch(config)#ip arp inspection validate dst-mac`	Configures DAI to drop ARP packets when the destination MAC address in the body of the ARP packet does not match the destination MAC address specified in the Ethernet header. This check is performed on both ARP requests and responses.
`Switch(config)#ip arp inspection validate ip`	Configures DAI to drop ARP packets that have invalid and unexpected IP addresses in the ARP body, such as 0.0.0.0, 255.255.255.255, or all IP multicast addresses. Sender IP addresses are checked in all ARP requests and responses, and target IP addresses are checked only in ARP responses.
`Switch(config)#interface fastethernet0/24`	Moves to interface configuration mode.
`Switch(config-if)#ip dhcp snooping trust`	Configures the interface as trusted for DHCP snooping.
`Switch(config-if)#ip arp inspection trust`	Configures the connection between switches as trusted for DAI.
	NOTE: By default, all interfaces are untrusted.

TIP: It is generally advisable to configure all access switch ports as untrusted and to configure all uplink ports that are connected to other switches as trusted.

Verifying DAI

`Switch#show ip arp inspection interfaces`	Verifies the dynamic ARP configuration
`Switch#show ip arp inspection vlan 10`	Verifies the dynamic ARP configuration for VLAN 10
`Switch#show ip arp inspection statistics vlan 10`	Displays the dynamic ARP inspection statistics for VLAN 10

Mitigating VLAN Hopping: Best Practices

Configure all unused ports as access ports so that trunking cannot be negotiated across those links.

Place all unused ports in the shutdown state and associate with a VLAN designed only for unused ports, carrying no user data traffic.

When establishing a trunk link, purposefully configure the following:

- The native VLAN to be different from any data VLANs
- Trunking as **on**, rather than **negotiated**
- The specific VLAN range to be carried on the trunk (prune the native VLAN from the allowed VLAN list)

NOTE: Maintenance protocols, such as Cisco Discovery Protocol (CDP) and Dynamic Trunking Protocol (DTP), are normally carried over the native VLAN. Native VLAN pruning will not affect them; they will still communicate on a pruned native VLAN.

TIP: It is also possible to tag all VLANs, including the native VLAN. This is done with the global configuration command **vlan dot1q tag native**.

VLAN Access Lists

VLAN ACLs (VACLs) can provide access control for all packets that are bridged within a VLAN or that are routed into or out of a VLAN or a WAN interface for VACL capture. Unlike Cisco IOS ACLs that are applied on routed packets only, VACLs apply to all packets and can be applied to any VLAN or WAN interface. VLAN access maps do not work on the Catalyst 2960 switch platform, but they do work on the Catalyst 3560, 3750, and the 6500 switch platforms.

NOTE: VACLs have an implicit deny at the end of the map; a packet is denied if it does not match any ACL entry, and at least one ACL is configured for the packet type.

Figure 14-1 shows the order in which packets are filtered by PACLs, VACLs, and traditional IOS ALCs.

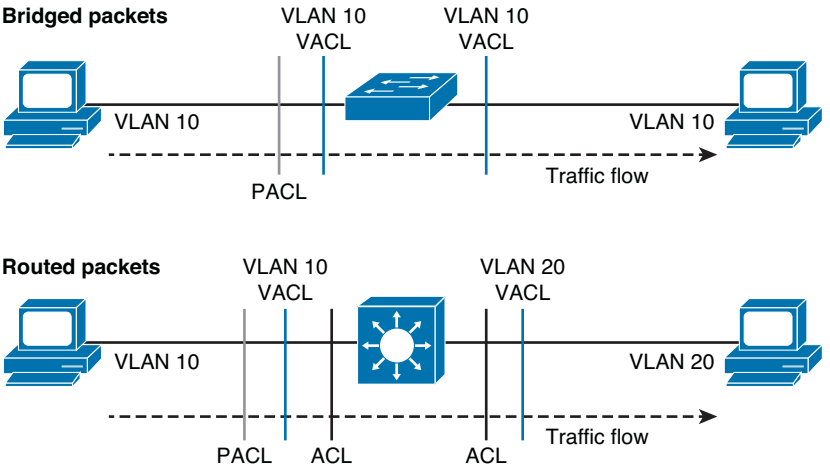


Figure 14-1 Interaction Between PACLs, VACLs, and IOS ACLs

`Switch(config)#ip access-list extended TEST1`	Creates a named extended ACL called TEST1.
`Switch(config-ext-nacl)#permit tcp any any`	The first line of the extended ACL will permit any TCP packet from any source to travel to any destination address. Because there is no other line in this ACL, the implicit deny statement that is part of all ACLs will deny any other packet.
`Switch(config-ext-nacl)#exit`	Exits named ACL configuration mode and returns to global config mode.
`Switch(config)#mac access-list extended SERVER2`	Create the extended MAC access-list SERVER2.
`Switch(config-ext-mac)#permit any host 0000.1111.2222`	Permit traffic from any source to the destination specified by the MAC address 0000.1111.2222
	NOTE: Because the access list will be used in the access map DROP1, the "permit" statement in the MAC access list is not permitting this traffic but rather choosing the traffic that will be acted upon in the action portion of the access map.
`Switch(config)#vlan access-map DROP1 5`	Creates a VLAN access map named DROP1 and moves into VLAN access map configuration mode. A sequence number of 5 is assigned to this access map. If no sequence number is given at the end of the command, a default number of 10 is assigned.
`Switch(config-access-map)#match ip address TEST1`	Defines what needs to occur for this action to continue. In this case, packets filtered out by the named ACL TEST1 will be acted upon.
	NOTE: You can match ACLs based on the following: IP ACL number: 1–199 and 1300–2699 IP ACL name MAC address ACL name
`Switch(config-access-map)#action drop`	Any packet that is filtered out by the ACL TEST1 will be dropped.
	NOTE: You can configure the following actions: Drop Forward Redirect (works only on a Catalyst 6500)

`Switch(config)#vlan access-map DROP1 10`	Creates line 10 of the VLAN access map named DROP1.
`Switch(config-access-map)#match mac address SERVER2`	Matches the MAC access list filter SERVER2.
`Switch(config-access-map)#action drop`	Drops all traffic permitted by the MAC access list SERVER2.
`Switch(config-access-map)#vlan access-map DROP1 15`	Creates line 15 of the VLAN access map named DROP1.
`Switch(config-map)#action forward`	Forwards traffic not specified to be dropped in line 5 and 10 of the VLAN access map DROP1.
`Switch(config-access-map)#exit`	Exits access map configuration mode.
`Switch(config)#vlan filter DROP1 vlan-list 20-30`	Applies the VLAN map named DROP1 to VLANs 20–30.
	NOTE: The **vlan-list** argument can refer to a single VLAN (26), a consecutive list (20–30), or a string of VLAN IDs (12, 22, 32). Spaces around the comma and hyphen are optional.

Verifying VACLs

`Switch#show vlan access-map`	Displays all VLAN access maps
`Switch#show vlan access-map DROP1`	Displays the VLAN access map named DROP1
`Switch#show vlan filter`	Displays what filters are applies to all VLANs
`Switch#show vlan filter access-map DROP1`	Displays the filter for the specific VLAN access map named DROP1

Configuration Example: VACLs

Figure 14-2 shows the network topology for the configuration that follows, which demonstrates how to configure VACLs using the commands covered in this chapter.

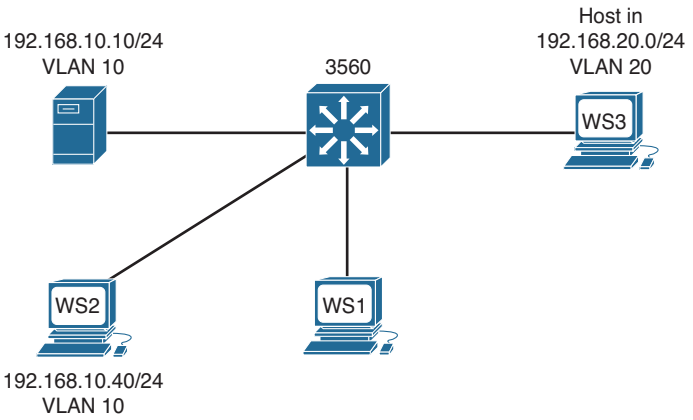


Figure 14-2 Network Topology for VACL Configuration

The objective of the VACL is to deny all IP traffic from VLAN 20 from reaching the server in VLAN 10. A specific host in VLAN 10 with an IP address of 192.168.10.40/24 is also denied access to the server. All other IP traffic is allowed. A 3560 switch is used for this example.

`3560(config)#ip access-list extended DENY_SERVER_ACL`	Creates a named ACL called DENY_SERVER_ACL and moves to named ACL configuration mode.
`3560(config-ext-nacl)#permit ip 192.168.20.0 0.0.0.255 host 192.168.10.10`	This line filters out all IP packets from a source address of 192.168.20.*x* destined for the server at 192.168.10.10.
`3560(config-ext-nacl)#permit ip host 192.168.10.40 host 192.168.10.10`	This line filters out all IP packets from a source address of 192.168.10.40 destined for the server at 192.168.10.10.
`3560(config-ext-nacl)#exit`	Returns to global configuration mode.
`3560(config)#vlan access-map DENY_SERVER_MAP 10`	Creates a VACL called DENY_SERVER_MAP and moves into VLAN access map configuration mode. If no sequence number is given at the end of the command, a default number of 10 is assigned.
`3560(config-access-map)#match ip address DENY_SERVER_ACL`	Defines what needs to occur for this action to continue. In this case, packets filtered out by the named ACL DENY_SERVER_ACL will be acted upon.
`3560(config-access-map)#action drop`	Any packet filtered out by the ACL will be dropped
`3560(config-access-map)#exit`	Returns to global configuration mode
`3560(config)#vlan access-map DENY_SERVER_MAP 20`	Creates line 20 of the VACL called DENY_SERVER_MAP and moves into VLAN access map configuration mode

`3560(config-access-map)#action forward`	Any other packets not filtered out by the ACL in line 10 will be forwarded since there is no specific **match** statement.
`3560(config-access-map)#exit`	Returns to global configuration mode
`3560(config)#vlan filter DENY_SERVER_MAP vlan-list 10`	Applies the VACL to VLAN 10

Private VLANs

A private VLAN (PVLAN) partitions the Layer 2 broadcast domain of a VLAN into subdomains, thus isolating the ports on the switch from each other, while keeping them in the same subnet. A PVLAN is essentially a VLAN inside a VLAN all sharing the same IP subnet.

> **NOTE:** Private VLANs are implemented to varying degrees on Catalyst 6500/4500/3750/3560 as well as the Metro Ethernet line of switches. All PVLAN configuration commands are not supported on all switch platforms. For more information, see Appendix A, "Private VLAN Catalyst Switch Support Matrix."

A PVLAN domain has one primary VLAN. Each port in a PVLAN domain is a member of the primary VLAN. Secondary VLANs are subdomains that provide isolation between ports within the same PVLAN domain. There are two types of secondary VLANs: isolated VLANs and community VLANs. Isolated VLANs contain isolated ports. Community VLANs contain community ports. A port that belongs to the primary VLAN and can communicate with all mapped ports in the primary VLAN, including community and isolated ports, is called a promiscuous port.

`Switch(config)#vtp mode transparent`	Sets VLAN Trunking Protocol (VTP) mode to transparent. This is a requirement before configuring PVLANs.
`Switch(config)#vlan 20`	Creates VLAN 20 and moves to VLAN configuration mode.
`Switch(config-vlan)#private-vlan primary`	Creates a private, primary VLAN.
`Switch(config-vlan)#vlan 101`	Creates VLAN 101 and moves to VLAN configuration mode.
`Switch(config-vlan)#private-vlan isolated`	Creates a private, isolated VLAN for VLAN 101.
	NOTE: An isolated VLAN can only communicate with promiscuous ports. Ports within an isolated VLAN cannot communicate with each other at the Layer 2 level but can still communicate with a promiscuous port.
`Switch(config-vlan)#exit`	Returns to global configuration mode.

`Switch(config)#vlan 102`	Creates VLAN 102 and moves to VLAN configuration mode.
`Switch(config-vlan)#private-vlan community`	Creates a private, community VLAN for VLAN 102.
	NOTE: A community VLAN can communicate with all promiscuous ports and with other ports in the same community.
`Switch(config-vlan)#exit`	Returns to global configuration mode.
`Switch(config)#vlan 103`	Creates VLAN 103 and moves to VLAN configuration mode.
`Switch(config-vlan)#private-vlan community`	Creates a private, community VLAN for VLAN 103.
`Switch(config-vlan)#vlan 20`	Returns to VLAN configuration mode for VLAN 20.
`Switch(config-vlan)#private-vlan association 101-103`	Associates secondary VLANs 101–103 with primary VLAN 20
	NOTE: Only one isolated VLAN can be mapped to a primary VLAN, but more than one community VLAN can be mapped to a primary VLAN.
`Switch(config)#interface fastethernet0/20`	Moves to interface configuration mode.
`Switch(config-if)#switchport mode private-vlan host`	Configures the port as a private VLAN host port.
`Switch(config-if)#switchport private-vlan host-association 20 101`	Associates the port with primary private VLAN 20 and secondary private VLAN 101.
`Switch(config-if)#interface fastethernet0/21`	Moves to interface configuration mode.
`Switch(config-if)#switchport mode private-vlan promiscuous`	Configures the port as a private VLAN promiscuous port.
`Switch(config-if)#switchport private-vlan mapping 20 101-103`	Associates the port with primary private VLAN 20 and secondary private VLAN 101.

Verifying PVLANs

`Switch#show vlan private-vlan [type]`	Verifies private VLAN configuration
`Switch#show interface fastethernet0/20 switchport`	Verifies all configuration on FastEthernet0/20, including private VLAN associations

NOTE: It is possible to configure special trunking for PVLAN support. This functionality is only available on the Catalyst 4500 and 6500 series modular switches. See the following document for additional information: http://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst4500/12-2/54sg/configuration/guide/config/pvlans.html.

Configuration Example: PVLANs

Figure 14-3 shows the network topology for the configuration that follows, which demonstrates how to configure PVLANs using the commands covered in this chapter. The following network functionality is required:

- All ISP clients A, B, and C are in the same primary VLAN, same subnet.
- Customer A locations can only exchange data between each other and can access the ISP router.
- Customer B locations can only exchange data between each other and can access the ISP router.
- Customer C can only exchange data with the ISP router.
- SW1 and SW2 operate at Layer 2 only. Routing occurs at ISP router.

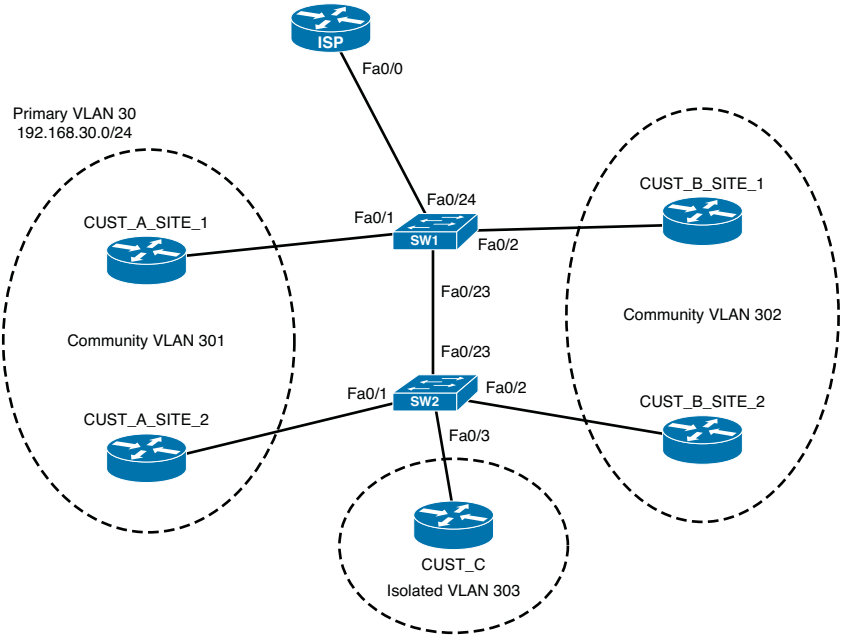


Figure 14-3 Network Topology for PVLAN Configuration Example

Switch SW1

`SW1(config)#vtp mode transparent`	Specifies the VTP device mode as transparent
`SW1(config)#vlan 301`	Creates VLAN 301
`SW1(config-vlan)#private-vlan community`	Defines the VLAN as private with community ports
`SW1(config-vlan)#vlan 302`	Creates VLAN 302

`SW1(config-vlan)#private-vlan community`	Defines the VLAN as private with community ports
`SW1(config-vlan)#vlan 303`	Creates VLAN 303
`SW1(config-vlan)#private-vlan isolated`	Defines the VLAN as private with isolated ports
`SW1(config-vlan)#vlan 30`	Creates VLAN 30
`SW1(config-vlan)#private-vlan primary`	Defines the VLAN as the primary VLAN for the private VLANs
`SW1(config-vlan)#private-vlan association 301-303`	Associates the secondary VLANs with the primary VLAN 30
`SW1(config-vlan)#interface fastethernet0/1`	Moves to interface configuration mode
`SW1(config-if)#switchport private-vlan association 30 301`	Defines the port as private with a primary VLAN of 30 and a secondary (community) VLAN of 301
`SW1(config-if)#switchport mode private-vlan host`	Configures the interface as a PVLAN host port
`SW1(config-if)#interface fastethernet0/2`	Moves to interface configuration mode
`SW1(config-if)#switchport private-vlan association 30 302`	Defines the port as private with a primary VLAN of 30 and a secondary (community) VLAN of 302
`SW1(config-if)#switchport mode private-vlan host`	Configures the interface as a PVLAN host port
`SW1(config-if)#interface fastethernet0/23`	Moves to interface configuration mode
`SW1(config-if)#switchport trunk encapsulation dot1q`	Sets the interface to an 802.1Q trunk
`SW1(config-if)#switchport mode trunk`	Sets the port to trunk unconditionally
`SW1(config-if)#interface fastethernet0/24`	Moves to interface configuration mode
`SW1(config-if)#switchport trunk encapsulation dot1q`	Sets the interface to an 802.1Q trunk
`SW1(config-if)#switchport mode trunk`	Sets the port to trunk unconditionally
`SW1(config-if)#switchport mode private-vlan promiscuous`	Configures the interface as a PVLAN promiscuous port
`SW1(config-if)#switchport private-vlan mapping 30 301-303`	Maps the primary and secondary VLANs to the promiscuous port

Switch SW2

SW2(config)#**vtp mode transparent**	Specifies the VTP device mode as transparent
SW2(config)#**vlan 301**	Creates VLAN 301
SW2(config-vlan)#**private-vlan community**	Defines the VLAN as private with community ports
SW2(config-vlan)#**vlan 302**	Creates VLAN 302
SW2(config-vlan)#**private-vlan community**	Defines the VLAN as private with community ports
SW2(config-vlan)#**vlan 303**	Creates VLAN 303
SW2(config-vlan)#**private-vlan isolated**	Defines the VLAN as private with isolated ports
SW2(config-vlan)#**vlan 30**	Creates VLAN 30
SW2(config-vlan)#**private-vlan primary**	Defines the VLAN as the primary VLAN for the private VLANs
SW2(config-vlan)#**private-vlan association 301-303**	Associates the secondary VLANs with the primary VLAN 30
SW2(config-vlan)#**interface fastethernet0/1**	Moves to interface configuration mode
SW2(config-if)#**switchport private-vlan association 30 301**	Defines the port as private with a primary VLAN of 30 and a secondary (community) VLAN of 301
SW2(config-if)#**switchport mode private-vlan host**	Configures the interface as a private-VLAN host port
SW2(config-if)#**interface fastethernet0/2**	Moves to interface configuration mode.
SW2(config-if)#**switchport private-vlan association 30 302**	Defines the port as private with a primary VLAN of 30 and a secondary (community) VLAN of 302.
SW2(config-if)#**switchport mode private-vlan host**	Configures the interface as a private-VLAN host port.
SW2(config-if)#**interface fastethernet0/3**	Moves to interface configuration mode.
SW2(config-if)#**switchport private-vlan association 30 303**	Defines the port as private with a primary VLAN of 30 and a secondary (isolated) VLAN of 303.
SW2(config-if)#**switchport mode private-vlan host**	Configures the interface as a private-VLAN host port.
SW2(config-if)#**interface fastethernet0/23**	Moves to interface configuration mode.
SW2(config-if)#**switchport trunk encapsulation dot1q**	Sets the interface to an 802.1Q trunk.
SW2(config-if)#**switchport mode trunk**	Sets the port to trunk unconditionally.

This page intentionally left blank

APPENDIX A

Private VLAN Catalyst Switch Support Matrix

Private VLANs (PVLAN) provide Layer 2 (L2) isolation between ports in the same VLAN. The table summarizes the support of the PVLAN feature in the Cisco Catalyst switches.

Catalyst Platform	**PVLAN Supported Minimum Software Version**	**Isolated VLAN**	**PVLAN Edge (Protected Port)**	**Community VLAN**
Catalyst 6500/6000 - Hybrid mode (CatOS on Supervisor and Cisco IOS on MSFC)	5.4(1) on Supervisor and 12.0(7)XE1 on MSFC	Yes	Not supported	Yes
Catalyst 6500/6000 - Native mode (Cisco IOS System software on both Supervisor and MSFC)	12.1(8a)EX, 12.1(11b)E1 and later.	Yes	Not supported	Yes
Catalyst 4500/4000 - CatOS	6.2(1)	Yes	Not supported	Yes
Catalyst 4500/4000 - Cisco IOS	12.1(8a)EW	Yes	Not Supported	Yes, 12.2(20) EW onwards
Catalyst 4500-X, Catalyst 4500-E	All	Yes	Not supported	Yes
Catalyst 3550	All	Not Supported	Yes. 12.1(4) EA1 onwards.	Not Supported
Catalyst 2950	All	Not Supported	Yes. 12.0(5.2) WC1, 12.1(4) EA1 and later.	Not Supported
Catalyst 3560	12.2(20)SE - EMI	Yes	Yes. 12.1(19) EA1 onwards.	Yes
Catalyst 3750	12.2(20)SE - EMI	Yes	Yes. 12.1(11) AX onwards.	Yes
Catalyst 3750 Metro	12.2(25)EY - EMI	Yes	Yes. 12.1(14) AX onwards.	Yes
Catalyst 2960	All	Not Supported	Yes. 12.2(25) FX and later.	Not Supported

Catalyst Platform	PVLAN Supported Minimum Software Version	Isolated VLAN	PVLAN Edge (Protected Port)	Community VLAN
Catalyst 2960-S Catalyst 2960-C Catalyst 2960-X	15.2(1)E	Not Supported	Not Supported	Not Supported
Catalyst Express 500	All	Not Supported	Not Supported	Not Supported
Nexus 7000	NX-OS	Yes	Not Supported	Yes
Catalyst 6800	Release 15.1SY Sup Engine 2T	Yes	Not Supported	Yes
Catalyst 3850	Cisco IOS XE 3.3SE	Yes	Not Supported	Yes
Catalyst 3650	All	Not Supported	Not Supported	Not Supported
Metro Ethernet 3400	12.(60)EZ	Yes	Not Supported	Yes

APPENDIX B

Create Your Own Journal Here

Even though we have tried to be as complete as possible in this reference guide, invariably we will have left something out that you need in your specific day-to-day activities. That is why this section is here. Use these blank lines to enter in your own notes, making this reference guide your own personalized journal.

Index

Numbers

A

B

C

D

E

F

I

L

M

N

O

P

R

S

T

U

V

W - X - Y - Z

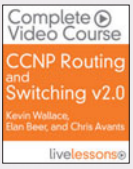
This page intentionally left blank

Cisco Press

NEW Complete Video Courses for CCNP Routing & Switching 300 Series Exams

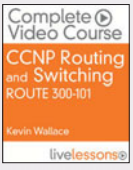
These unique products include multiple types of video presentations, including:

- Live instructor whiteboarding
- Real-world demonstrations
- Animations of network activity
- Dynamic KeyNote presentations
- Doodle videos
- Hands-on command-line interface (CLI) demonstrations
- Review quizzes



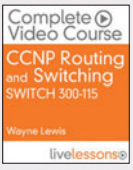
9780789754493

CCNP Routing and Switching v2.0 – Complete Video Course Library
Specially priced library including ALL THREE Complete Video Courses: *CCNP Routing and Switching ROUTE 300-101*, *CCNP Routing and Switching SWITCH 300-115*, and *CCNP Routing and Switching TSHOOT 300-135*.



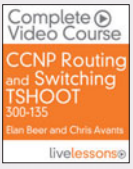
9780789753731

CCNP Routing and Switching ROUTE 300-101 – Complete Video Course
149 VIDEOS with 12+ HOURS of video instruction from best-selling author, expert instructor, and double CCIE Kevin Wallace walk you through the full range of topics on the CCNP Routing and Switching ROUTE 300-101 exam, including fundamental routing concepts; IGP routing protocols including RIPng, EIGRP, and OSPF; route distribution and selection; BGP; IPv6 Internet connectivity; router security; and routing protocol authentication.



9780789754073

CCNP Routing and Switching SWITCH 300-115 – Complete Video Course
10+ HOURS of unique video training walks you through the full range of topics on the CCNP SWITCH 300-115 exam. This complete video course takes you from the design and architecture of switched networks through the key technologies vital to implementing a robust campus network. You will learn, step-by-step, configuration commands for configuring Cisco switches to control and scale complex switched networks.



9780789754295

CCNP Routing and Switching TSHOOT 300-135 – Complete Video Course
10+ HOURS of unique video instruction from expert instructors and consultants Elan Beer and Chris Avants walks you through the full range of topics on the CCNP TSHOOT 300-135 exam. This complete video course teaches you the skills you need to plan and perform regular maintenance on complex enterprise routed and switched networks and how to use technology-based practices and a systematic ITIL-compliant approach to perform network troubleshooting commands for configuring Cisco switches to control and scale complex switched networks.

SAVE ON ALL NEW

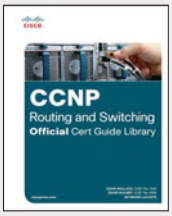
CCNP R&S 300 Series Products

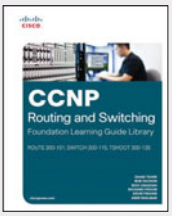
www.CiscoPress.com/CCNP

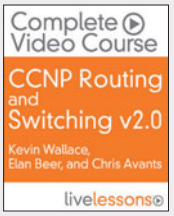
Cisco Press

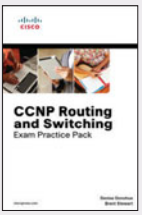
NEW Learning Materials for CCNP Routing & Switching 300 Series Exams

Increase learning, comprehension, and certification readiness with these Cisco Press products!









Complete Exam Preparation

Official Certification Guides

Each Official Cert Guide includes a test preparation routine proven to help you pass the exams, two practice tests with thorough exam topic reviews, hundreds of questions, a study plan template, unique review exercises like mind maps and memory tables, and much more.

Official Certification Guide Premium Editions

Digital-only products combining an Official Cert Guide eBook with additional exams in the Pearson IT Certification Practice Test engine.

Complete Video Courses

Real-world demonstrations, animations, configuration walkthroughs, whiteboard instruction, dynamic presentations, and live instruction bring Cisco CCNP ROUTE, SWITCH, and TSHOOT exam topics to life.

Foundation Learning Guides

Provide early and comprehensive foundation learning for the new CCNP exams. These revisions to the popular Authorized Self-Study Guide format are fully updated to include complete coverage.

Late Stage Preparation and Reference

Quick References

As a final preparation tool, these provide you with detailed, graphical-based information, highlighting only the key topics on the latest CCNP exams in cram-style format.

Cert Flash Cards Online

This online exam preparation tool consists of a custom flash card application loaded with 300 questions that test your skills and enhance retention of exam topics.

Portable Command Guide

Summarizes all CCNP certification-level Cisco IOS Software commands, keywords, command arguments, and associated prompts.

SAVE ON ALL NEW

CCNP R&S 300 Series Products

Plus **FREE SHIPPING** in the U.S. at **www.CiscoPress.com/CCNP**