Official Cert Guide

Advance your IT career with hands-on learning

CCNP Collaboration Call Control and Mobility

CLACCM 300-815

Kyzer Davis, CCIE® No. 54735
Paul Giralt, CCIE® No. 4793
Patrick Kinane, CCIE® No. 58284
Gonzalo Salgueiro, CCIE® No. 4541

ciscopress.com

CCNP Collaboration Call Control and Mobility

CLACCM 300-815

Official Cert Guide

Enhance Your Exam Preparation

Save 80% on Premium Edition eBook and Practice Test

The *CCNP Collaboration Call Control and Mobility CLACCM 300-815 Official Cert Guide Premium Edition and Practice Test* provides three eBook files (PDF, EPUB, and MOBI/Kindle) to read on your preferred device and an enhanced edition of the Pearson Test Prep practice test software. You also receive two additional practice exams with links for every question mapped to the PDF eBook.

CCNP Collaboration Call Control and Mobility

CLACCM 300-815

Official Cert Guide

KYZER DAVIS, CCIE No. 54735
PAUL GIRALT, CCIE No. 4793
PATRICK KINANE, CCIE No. 58284
GONZALO SALGUEIRO, CCIE No. 4541

Cisco Press

CCNP Collaboration Call Control and Mobility CLACCM 300-815 Official Cert Guide

Kyzer Davis, Paul Giralt, Patrick Kinane, Gonzalo Salgueiro

Published by:
Cisco Press
Hoboken, NJ

ScoutAutomatedPrintCode

Library of Congress Control Number: 2020945984

ISBN-13: 978-0-13-657554-2

ISBN-10: 0-13-657554-4

Warning and Disclaimer

This book is designed to provide information about the CCNP Implementing Cisco Advanced Call Control and Mobility Services (CLACCM) 300-815 exam. 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 authors 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 intlcs@pearson.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.

Editor-in-Chief: Mark Taub

Alliances Manager, Cisco Press: Arezou Gol

Director, ITP Product Management: Brett Bartow

Executive Editor: Nancy Davis

Managing Editor: Sandra Schroeder

Development Editor: Ellie Bru

Senior Project Editor: Tonya Simpson

Copy Editor: Kitty Wilson

Technical Editors: Michael Mendoza Guzman, Esteban Valverde

Editorial Assistant: Cindy Teeters

Cover Designer: Chuti Prasertsith

Composition: codeMantra

Indexer: Timothy Wright

Proofreader: Donna Mulder

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

About the Authors

Kyzer Davis, CCIE Collaboration No. 54735, is an escalation point for various worldwide cloud and collaboration teams within Cisco Systems. He is the focal point for supporting, troubleshooting, and resolving complex, solution-level problems involving voice, video, cloud, and security portions of the Cisco Unified collaboration product portfolio. In addition to his work on this book, Kyzer has authored and presented numerous technical white papers on Cisco collaboration configuration, architecture, protocol design, and security topics, many of which are leveraged by Cisco partners, Cisco exam instructors, and Cisco employees for training and education.

Prior to writing this book, Kyzer co-authored the Cisco Press publication *Understanding Session Border Controllers: Comprehensive Guide to Designing, Deploying, Troubleshooting, and Maintaining Cisco Unified Border Element (CUBE) Solutions*. In addition, he works with Learning@Cisco on strategy and content development for numerous Cisco certifications. Kyzer is a technology enthusiast and mentor who is always working on automation initiatives and dabbling in new and evolving technology. He also enjoys a mean barbecue.

Paul Giralt, CCIE R&S, CCIE Voice, CCIE Collaboration No. 4793, is a distinguished engineer in the Customer Experience organization, where he focuses on Cisco collaboration technologies. He spends much of his time helping Cisco's largest customers accelerate the adoption of Cisco technologies and solutions and building new services capabilities to provide more proactive and predictive services as well as improve product serviceability. Paul has spent his 22+-year career at Cisco in TAC, the Collaboration Technology Group, Solution Validation Services, Cisco Advanced Services, and, most recently, as part of the Customer Experience organization. Paul has spent years troubleshooting and diagnosing issues on some of the largest and most complex Cisco collaboration deployments. He is passionate about the intersection of programmability and Cisco collaboration products as well as giving customers the knowledge and tools they need to diagnose issues on their own.

Paul is the author of the classic Cisco Press title *Troubleshooting Cisco IP Telephony* and is well known for his Cisco Live! sessions on SIP, collaboration APIs, troubleshooting Cisco collaboration, and other topics related to Cisco collaboration, making him a member of the Cisco Live! Distinguished Speaker Hall of Fame Elite. He is also the creator of the TranslatorX tool, which is used across the Cisco collaboration industry, as well as a contributor to SIP standards in the IETF. Paul has a degree in computer engineering from the University of Miami.

Patrick Kinane, CCIE Collaboration No. 58284, is a senior engineer and escalation point for the worldwide Cisco Unified Communications Manager teams within Cisco Technical Assistance Center (TAC). He is a subject matter expert (SME) for Cisco collaboration products such as Cisco Unified Communications Manager, Cisco collaboration

endpoints, Cisco Paging Server, and Cisco Emergency Responder. Patrick can be found on the following social media sites:

YouTube: https://www.youtube.com/patrick_kinane1

Twitter: https://twitter.com/patrick__k9

LinkedIn: https://www.linkedin.com/in/patrickkinane/

Gonzalo Salgueiro, CCIE No. 4541, is a distinguished engineer at Cisco working in the Customer Experience Technology Office (CXTO) helping to shape technical and organizational strategies, leading innovation and automation initiatives, and driving engineering excellence. Gonzalo has spent 23+ years at Cisco, establishing himself as a subject matter expert, an innovator, and an industry thought leader in various technologies, including collaboration, cloud, and IoT.

Gonzalo is an established member of numerous industry organizations and is a regular presenter and distinguished speaker at a variety of technical industry conferences and Cisco events around the world. He has held various industry leadership roles, including serving as a member of the board of directors of the SIP Forum, co-chair of the ASAP, INSIPID, and SIPBRANDY IETF working groups, a member of the IoT Directorate in the IETF, and co-chair of the WebRTC Task Group, IPv6 Task Group, and FoIP Task Group in the SIP Forum. He is an active contributor to various industry organizations and standardization activities.

Gonzalo previously co-authored three Cisco Press books on IoT and collaboration technologies. He has also co-authored 25 IETF RFCs, 4 IEEE papers, 4 ITU contributions, and numerous industry and academic research papers on a variety of different technical topics. He is also co-inventor of 120+ patents (issued and pending) and has contributed to various interop and open source development efforts. Gonzalo received a master's degree in physics from the University of Miami.

NOTE In addition to all the authors on this book holding a CCIE-level certification, both Kyzer Davis and Patrick Kinane hold a CCNP Specialist Certification for the concentration exam Implementing Cisco Advanced Call Control and Mobility Services (CLACCM) 300-815.

About the Technical Reviewers

Esteban Valverde, CCIE No. 34305, is a Cisco technical leader with 14 years of experience in networking and software development. In his current role, he is part of the Cisco collaboration team of the Technical Assistance Center (TAC) in Research Triangle Park, North Carolina. His main focus is working with customers and Cisco support engineers on escalations that involve Cisco Unified Border Element (CUBE), fax and modems, Cisco Unified Communications Manager, and Unified Contact Center Enterprise Deployments, as well as working closely with the Cisco engineering team to enhance product serviceability and automating common troubleshooting tasks. During the past years, he has specialized in troubleshooting complex issues for some of the largest VoIP networks and has provided technical leadership for some of the most critical worldwide collaboration deployments. Esteban has developed and delivered all levels of training and documentation on the Cisco Unified Border Element to Cisco technical teams. He holds a bachelor's degree in systems engineering.

Michael Mendoza, CCIE No. 34300, is based in Cisco's Research Triangle Park, North Carolina, office. He has worked on Unified Collaboration technologies for 12 years as part of the Multiservice (MS) and the Unified Communication Manager Technical Assistance Center (TAC) teams. In his current role as a technical leader within TAC, he is responsible for an escalation point as well as a point of contact for any technical support topics between TAC and the Cisco engineering team for multiple unified collaboration products. Michael's areas of expertise include Cisco Unified Communications Manager, endpoints, Cisco Emergency Responder, voice gateways, Cisco Unified Border Element (CUBE), and more. Michael is a seasoned Cisco Live! speaker and is heavily involved in automation, software development, and innovation projects; he is always striving to find new ways to make the lives of TAC engineers supporting Cisco customers globally easier.

Dedications

To my parents and grandparents, for encouraging me to always ask questions, explore new ideas, and expand my horizons by continuously striving to learn new topics throughout my career in information technology. Without their wisdom and guidance, I would not be where I am today.

—Kyzer

To my beautiful wife, who has always supported me and our family. Each time I work on a project, she takes on more of the tasks at home (especially when I was chasing the CCIE), and I appreciate all she does. To my amazing children, who help motivate me to be a better version of myself. To my parents, siblings, and extended family who all helped me grow up to be the man I am today: I love each of you very much.

Last, but not least, to all the people who've made the Cisco collaboration content and from whom I've learned over the years. Here is a short list: Vik Malhi, Mark Snow, Kevin Wallace, Paul Giralt, Jeremy Cioara, Andy Vassar, Ralph Smith III, and Network Chuck.

—Patrick

This book is affectionately dedicated to my family.

To my brilliant and beautiful wife, Becky, whose infinite patience and steadfast love, friendship, and support have been the inspirational driving force in my life.

To our wonderful children, Alejandro, Sofia, Gabriela, and Mateo, who have given me the greatest joy and have loved me unconditionally and without question all the days of their lives. It's hard to put into words how much I love all of you and how proud I am to be your father.

Finally, I dedicate this book to my parents, Alberto and Elena, who have given me so much for so long. I know that I will never be able to repay the lifetime of kindness, encouragement, and learning. Please know that I walk in your footsteps, knowing full well that the amazing life that I lead is only possible because of you.

—Gonzalo

I dedicate this to my wife, Archana, and my children, Rohen and Maya, for always supporting me and putting up with mountains of IP Phones, video devices, and network gear found around the house enabling me to do things like write this book.

—Paul

Acknowledgments

From Kyzer Davis:
A big hearty thank you to the rest of the author team for contributing to this publication and sharing their in-depth knowledge surrounding the various components, products, and protocols that make up the Cisco Unified collaboration product portfolio.

A smaller personal thank you to Joseph House, Mohammad Manna, Dillon Brown, Irina Hristova, Daniel Alejandro Alvarado Vega, Tim Thurber, Joshua Raja, Julio Cascante, Matt Gross, Scott Kiewert, Donald Hohenstein, Matt Taber, Hazel Pringle, Luis Gonzalez Galindo, Josh Meadows, Kyle West, Chris Enterline, Jared Compiano, Christopher Hsu, and Lyle Gardner for reviewing the chapters of this book at various stages during publication and providing valuable feedback on topic comprehension, structure, and flow.

From Patrick Kinane:
Thank you to all the other authors, but especially Kyzer. I know Kyzer put a significant amount of time into this book on chapters he authored as well as the others. A big thank you to our technical editors, the people who read the chapters to provide feedback, and Cisco Press.

Thank you to the TLs, PEs, DSEs, and managers within the collaboration TAC community for always being excited about the projects TAC engineers are taking on. Thank you for encouraging us along the way and for your continued guidance.

From Gonzalo Salgueiro:
Thanks to all my co-authors for their dedication and passion for making this book a reality. Special thanks to Kyzer Davis for inviting me to contribute to this incredibly worthwhile endeavor. This is the second book we've worked on together, and seeing your growth as an engineer and a leader makes me proud.

Thanks to my management and leadership teams for their unwavering encouragement, belief, and support throughout this project. A nostalgic note of thanks to Marty Martinez, Marc Holloman, and Tom Berghoff for their friendship, their guidance, and the indelible impressions of leadership they imparted.

Thanks to all the technical editors and reviewers who have helped improve this manuscript. Emphatic thanks to Esteban Valverde and Michael Mendoza for the careful and thorough review of all the chapters to ensure that the content is technically accurate, useful, and easily consumable. This book is better because of Kaustubh Inamdar and his exceptional work on our recently published SBC book. I am immensely grateful to Kaustubh for his technical contributions and leadership.

A hearty thanks to all the customers, developers, engineers, and technical writers whom I have collaborated and co-innovated with during my many years at Cisco. Your generosity and willingness to share have made me a better engineer and have made projects like this book so rewarding.

Finally, thank you to everyone at Cisco Press for all the support with everything that happens after the technical words hit the page. We are grateful for all your efforts in

making us look good! Special thanks to our development editor, Ellie Bru, who is a shining example of patience, professionalism, and excellence. This our fourth book together, and I'm looking forward to many more.

From Paul Giralt:
Thank you to Kyzer for coming up with the idea to write this book and having the dedication to keep everything moving along to get it to the finish line. Also thank you to the other authors, who have been incredibly passionate about producing the best quality publication possible. It's been a pleasure working with all of you.

I'd also like to thank all our customers, who have supported our journey from the first IP phone more than 20 years ago. For those of you I have had the pleasure of working with, who read my previous books or attended Cisco Live sessions to learn more about how you can design, deploy, and troubleshoot Cisco's collaboration portfolio, we couldn't have gotten here without you. Keep challenging us to improve and evolve.

Thanks to all the reviewers for your excellent feedback, and to the Cisco Press team, who made the authoring process seamless. All your work makes us look good, and I appreciate the dedication and attention to detail.

Contents at a Glance

Reader Services

Other Features

In addition to the features in each of the core chapters, this book has additional study resources on the companion website, including the following:

Practice exams: The companion website contains an exam engine that enables you to review practice exam questions. Use these to prepare with a sample exam and to pinpoint topics where you need more study.

Interactive exercises and quizzes: The companion website contains interactive hands-on exercises and quizzes so that you can test your knowledge on the spot.

Key Term flash cards: The companion website contains interactive quizzes that enable you to test yourself on every glossary term in the book.

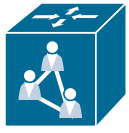
To access this additional content, simply register your product. To start the registration process, go to www.ciscopress.com/register and log in or create an account.* Enter the product ISBN 9780136575542 and click Submit. After the process is complete, you will find any available bonus content under Registered Products.

*Be sure to check the box that you would like to hear from us to receive exclusive discounts on future editions of this product.

Contents

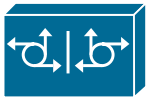
Icons Used in This Book

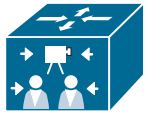

Phone/IP Phone


Media Control Unit


Cisco Instant Message and Presence


Enterprise


Unified Border Element


Cisco Video Communications Server Control


Server


Firewall

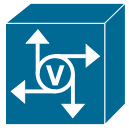
Unified CM


Unified CM

SIP Proxy Server


Cisco SRST Manager


Unified CM


Voice Gateway


Router


Telephony Router


Cisco Prime Collaboration

Voice Router


Cloud


Cisco Unity Connection

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

With the recent changes to the Cisco exam certifications, a professional-level certification is now more accessible than ever before. Within each Cisco Certified Network Professional (CCNP) track there now exists a single core exam and multiple concentration exams. To obtain a full CCNP Collaboration certification, one must pass the core exam and one concentration exam of choice within the CCNP Collaboration track. There are no longer CCNA prerequisites for the CCNP-level exams, and the CCNP concentration exams can be taken as standalone exams. For CCNP concentration exams, upon successful completion, a specialist certification is awarded. This added flexibility means that aspiring network engineers can tailor their learning and study habits specifically to what is directly required by their current position, company, or desired subject area. Furthermore, the core exam for each CCNP track now stands as the qualifying exam for the CCIE practical lab. Cisco professional certifications will continue to be an important part of the computing industry for many years to come.

Goals and Methods

The primary goal of this book is to help you pass the CLACCM (300-815) exam. With that in mind, we wanted to provide a Cisco Press publication that goes beyond what the exam blueprint details. After all, call routing and device mobility are very large parts of any Cisco Unified Communications network engineer's day-to-day operations. As you continue your Cisco collaboration journey, you will see that the topics covered in this book are relevant to many other features and integrations in the Cisco Unified CM product portfolio. This book is filled with Cisco best practices and deep dives into topics that will help network engineers day in and day out. This book has been written to a level that is accessible to a newcomer but continues to a level of knowledge that justifies keeping this book around as a reference even after you have passed your CCNP certification exam.

This book does not ask you to simply memorize topics in order to pass the exam. Instead, it explores the topics fully to provide a complete understanding of the subject at hand. We know that not every reader will be able to get hands-on lab experience before taking the exam. We therefore provide many hand-crafted figures illustrating scenarios and UI components. We also provide real-world relevant examples of output from CLI **show** commands, CLI **debug** commands, and trace/log files that are annotated and discussed at length within the text. These log examples serve two purposes. First, these examples help drive home the topics covered through actual call samples and log analysis. Second, these examples help you become familiar with some of the relevant snippets from working logs that you will one day leverage for on-the-job troubleshooting.

Who Should Read This Book?

This book is written for Cisco collaboration engineers who want to tremendously increase their chances of passing the CLACCM (300-815) CCNP Collaboration concentration exam. Whether you are taking this exam as a standalone exam for the specialist

certification, as part of a larger CCNP Collaboration journey, or even as part of a CCIE Collaboration journey, you can use this book to learn the information you need to know along with the technical depth required to implement, administer, and troubleshoot in real-world scenarios.

Strategies for Exam Preparation

There is no blanket strategy for exam preparation or for taking Cisco exams. The strategy you use depends on your level of existing knowledge about the topics at hand. For example, somebody who has been working with collaboration products for a few years might need to simply skim through the VoIP protocols chapters because on-the-job training and experience have provided a great foundation. Those who are new to the topics should review the VoIP protocols a few times to gain a full understanding of the topics described there. We highly encourage you not to entirely omit any topic area from your studies. Each chapter provides notes, tidbits, and information that might be new to you, and you may even learn about things you didn't know are possible.

Regardless of the strategy you use or the background you have, this book is designed to help you pass the exam in the least amount of time. Several book features will help you gain the confidence you need to be convinced that you know some material already and to also help you determine what topics you need to study more.

The Companion Website for Online Content Review

All the electronic review elements, as well as other electronic components of the book, exist on this book's companion website. To access the companion website, start by establishing a login at www.ciscopress.com and registering your book. To do so, simply go to www.ciscopress.com/register and enter the ISBN of the print book: 9780136575542. After you have registered your book, go to your account page and click the Registered Products tab. From there, click the Access Bonus Content link to get access to the book's companion website.

Note that if you buy the Premium edition eBook and Practice Test version of this book from Cisco Press, your book will automatically be registered on your account page. Simply go to your account page, click the Registered Products tab, and select Access Bonus Content to access the book's companion website.

How to Access the Pearson Test Prep Practice Test Software

You have two options for installing and using the Pearson Test Prep practice test software: a web app and a desktop app. To use the Pearson Test Prep application, start by finding the access code that comes with the book. You can find the code in these ways:

- **Print book:** Look in the cardboard sleeve in the back of the book for a piece of paper with your book's unique access code.
- **Premium edition:** If you purchase the Premium edition eBook and Practice Test directly from the Cisco Press website, the code will be populated on your account page after purchase. Just log in at www.ciscopress.com, click Account to see details of your account, and click the Digital Purchases tab.
- **Amazon Kindle:** For those who purchase a Kindle edition from Amazon, the access code will be supplied directly by Amazon.
- **Other bookseller eBooks:** Note that if you purchase an eBook version from any other source, the practice test is not included because other vendors to date have not chosen to vend the required unique access code.

NOTE Do not lose the access code because it is the only means with which you can access the QA content with the book.

Once you have the access code, to find instructions about both the Pearson Test Prep web app and the desktop app, follow these steps:

Step 1. Open this book's companion website.

Step 2. Click the Practice Exams button.

Step 3. Follow the instructions listed there for installing the desktop app and for using the web app.

If you want to use the web app only at this point, navigate to www.pearsontestprep.com, establish a free login if you do not already have one, and register this book's practice tests using the access code you just found. The process should take only a couple of minutes.

NOTE Amazon eBook (Kindle) customers: It is easy to miss Amazon's email that lists your Pearson Test Prep access code. Soon after you purchase the Kindle eBook, Amazon should send an email. However, the email uses very generic text and makes no specific mention of PTP or practice exams. To find your code, read every email from Amazon after you purchase the book. Also do the usual checks for ensuring your email arrives, like checking your spam folder.

NOTE Other eBook customers: As of the time of publication, only the publisher and Amazon supply Pearson Test Prep access codes when you purchase their eBook editions of this book.

How This Book Is Organized

Although this book can be read cover-to-cover, it is designed to be flexible and allow you to easily move between chapters and sections of chapters to read just the material that you need to get to know better. Where required, forward and reverse references are included to extend and tie topics into large discussions.

At first glance, the chapters may seem long—and some of them are—but the book needs to fully explore the topics at hand. No expense was spared in terms of the content covered for each topic. Because there is no CCNA prerequisite for this exam, the content of this book both provides introductory knowledge and builds on that foundation by discussing the intermediate to advanced topics related to each blueprint item.

The following list provides an at-a-glance summary of the chapters:

- **Chapter 1, "Introduction to Unified Collaboration and Dial Plans":** This chapter provides a crash course in Cisco Unified CM architecture, components, and products. In addition, it features a discussion about dial plan design, using a globalized dial plan as an example.
- **Chapter 2, "VoIP Protocols: SIP and H.323":** This chapter provides a vendor-neutral protocol analysis of the Session Initiation Protocol (SIP), Session Description Protocol (SDP), and H.323 components such as H.225 and H.245. The foundational content of this chapter is leveraged in many other chapters throughout the book.
- **Chapter 3, "VoIP Protocols: RTP, RTCP, and DTMF Relay":** This chapter discusses the protocols used for transporting real-time data such as audio and video media across an IP network. This chapter analyzes Real-Time Transport Protocol (RTP) and Real-Time Transport Control Protocol (RTCP), along with different types of DTMF relay methods used by endpoints to convey digit information over audio streams, RTP, SIP, and H.323.
- **Chapter 4, "Unified CM Call Routing and Digit Manipulation":** This chapter provides an in-depth explanation of Unified CM pattern matching, transformations, and masks. You will gain an understanding of device selection and the logical grouping concepts related to calling search spaces (CSS) and partitions. Dial plan elements such as tail-end hop off (TEHO) and intercluster Global Dial Plan Replication (GDPR) are discussed. The chapter ends with valuable dial plan troubleshooting steps, using various Cisco tools.

- **Chapter 5, "Unified CM SIP Trunk Configuration":** This chapter discusses Unified CM SIP trunking configuration, verification, and troubleshooting. It discusses general configurations as well as many advanced SIP trunk features and settings to provide insight into why you would enable or disable specific settings and features for a particular deployment or integration.
- **Chapter 6, "Unified CM Call Control Features":** This chapter discusses media resources, with a focus on the many conferencing resources available in Cisco collaboration deployments. Furthermore, the chapter discusses supplementary features such as call park and call pickup, along with regions, codec preference, and call admission control in Unified CM.
- **Chapter 7, "Unified CM Mobility":** This chapter discusses the configuration, verification, and troubleshooting of Unified CM mobility, which can be broadly broken down into three key areas: unified mobility, extension mobility, and device mobility.
- **Chapter 8, "CUBE Call Routing and Digit Manipulation":** This chapter provides an in-depth explanation of IOS, IOS XE, and CUBE call routing topics, including call legs, call flows, inbound/outbound dial peer matching, and application signaling and media binding. The chapter also discusses various digit, header, and URI manipulation techniques.
- **Chapter 9, "CUBE Interworking Features":** This chapter examines how CUBE performs SIP–SIP interworking as a back-to-back user agent (B2BUA) for sessions between Unified CM, SIP service providers, and other third-party SIP call control systems. This chapter discusses both initial call setup and mid-call signaling transactions, such as hold, resume, transfer, session refresh, and other generic session updates. This chapter also examines additional topics such as SIP toll fraud authentication on CUBE, service provider authentication, and CUBE media interworking and troubleshooting.
- **Chapter 10, "Unified CME and SRST":** This chapter discusses the initial configuration of Unified CME on IOS and IOS XE gateways to facilitate SIP IP phone registration. In addition, this chapter discusses advanced features such as voice hunt groups, multicast paging, and call park. Finally, it provides an overview of SRST with Unified CM.

Exam Topics

The questions on each certification exam are a closely guarded secret. However, Cisco has published exam blueprints that list which topics you must know to successfully complete the exam. Table I-1 lists the blueprint topics along with references to the book chapter or chapters where you can find more information about each topic. These are the same topics you should be proficient in when designing and implementing Cisco Unified CM networks in the real world. Note that some topics are discussed in multiple chapters as they pertain to specific devices and their configurations for specific protocols.

Table I-1 CCNP CLACCM 300-815 Exam Topics and Chapter References

Exam Topic	Chapter(s) in Which Topic Is Covered
1.1 Troubleshoot these elements of a SIP conversation	2, 5, 6, 9
1.1.a Early media	2, 5, 9
1.1.b PRACK	9
1.1.c Mid-call signaling (hold/resume, call transfer, conferencing)	2, 6, 9
1.1.d Session timers	5, 9
1.1.e UPDATE	2, 9
1.2 Troubleshoot these H.323 protocol elements	2, 3, 9
1.2.a DTMF	3, 9
1.2.b Call set up and tear down	2, 9
1.3 Troubleshoot media establishment	3, 5, 6, 9
2.1 Configure Cisco Unified Communications Manager Express for SIP phone registration	10
2.2 Configure Cisco Unified CME dial plans	10
2.3 Implement toll fraud prevention	10
2.4 Configure these advanced Cisco Unified CME features	10
2.4.a Hunt groups	10
2.4.b Call park	10
2.4.c Paging	10
2.5 Configure SIP SRST gateway	10
3.1 Configure these Cisco Unified Border Element dial plan elements	8, 9
3.1.a DTMF	3, 9
3.1.b Voice translation rules and profiles	8
3.1.c Codec preference list	9
3.1.d Dial peers	8
3.1.e Header and SDP manipulation with SIP profiles	8
3.1.f Signaling and media bindings	8
3.2 Troubleshoot these Cisco Unified Border Element dial plan elements	8, 9
3.2.a DTMF	3, 9
3.2.b Voice translation rules and profiles	8
3.2.c Codec preference list	9
3.2.d Dial peers	8
3.2.e Header and SDP manipulation with SIP profiles	8
3.2.f Signaling and media bindings	8
4.1 Configure these globalized call routing elements in Cisco Unified Communications Manager	4, 5

Exam Topic	Chapter(s) in Which Topic Is Covered
4.1.a Translation patterns	4
4.1.b Route patterns	4
4.1.c SIP route patterns	4
4.1.d Transformation patterns	4
4.1.e Standard local route group	4
4.1.f TEHO	4
4.1.g SIP trunking	5
4.2 Troubleshoot these globalized call routing elements in Cisco Unified Communications Manager	4, 5
4.2.a Translation patterns	4
4.2.b Route patterns	4
4.2.c SIP route patterns	4
4.2.d Transformation patterns	4
4.2.e Standard local route group	4
4.2.f TEHO	4
4.2.g SIP trunking	5
5.1 Troubleshoot Call Admission Control (exclude RSVP)	6
5.2 Configure ILS, URI synchronization, and GDPR	4
5.3 Configure hunt groups	4
5.4 Configure call queuing	4
5.5 Configure time of day routing	4
5.6 Configure supplementary functions	6
5.6.a Call park	6
5.6.b Meet-me	6
5.6.c Call pick-up	6
6.1 Configure Cisco Unified Communications Manager Mobility	7
6.1.a Unified Mobility	7
6.1.b Extension Mobility	7
6.1.c Device Mobility	7
6.2 Troubleshoot Cisco Unified Communications Manager Mobility	7
6.2.a Unified Mobility	7
6.2.b Extension Mobility	7
6.2.c Device Mobility	7

Each version of the exam can have topics that emphasize different functions or features, and some topics can be rather broad and generalized. The goal of this book is to provide the most comprehensive coverage to ensure that you are well prepared for the exam.

Although some chapters might not address specific exam topics, they provide a foundation that is necessary for a clear understanding of important topics. Your short-term goal might be to pass this exam, but your long-term goal should be to become a qualified CCNP collaboration engineer.

It is important to understand that this book is a static reference, whereas the exam topics are dynamic. Cisco can and does change the topics covered on certification exams often.

This book should not be your only reference when preparing for the certification exam. You can find a wealth of information at Cisco.com that covers each topic in great detail. If you think you need more detailed information on a specific topic, read the Cisco documentation that focuses on that topic.

NOTE As CCNP Collaboration technologies continue to evolve, Cisco reserves the right to change the exam topics without notice. Although you can refer to the list of exam topics in Table I-1, always check Cisco.com to verify the actual list of topics to ensure that you are prepared before taking the exam. You can view the current exam topics on any current Cisco certification exam by visiting https://learningnetwork.cisco.com/s/claccm-exam-topics. Note also that, if needed, Cisco Press might post additional preparatory content on the web page associated with this book at www.ciscopress.com/title/9780136575542. It's a good idea to check the website a couple of weeks before taking your exam to be sure that you have up-to-date content.

Figure Credit

Figure 3-6, screenshot of Wireshark ptime example © Wireshark

CHAPTER 2

VoIP Protocols: SIP and H.323

This chapter includes the following main topics:

Overview of SIP: This section provides a brief history of SIP and describes its functional components, the different methods commonly used in SIP, and the process by which a call is set up and torn down.

Introduction to SDP: This section examines Session Description Protocol (SDP) fundamentals and describes the offer/answer framework.

Overview of H.323: This section covers H.323 basics, including the various components typically included in H.323 networks and the flow for a typical H.323 call.

This chapter covers the following CLACCM 300-815 exam topics:

- 1.1 Troubleshoot these elements of a SIP conversation
  - 1.1.a Early media
  - 1.1.c Mid-call signaling (hold/resume, call transfer, conferencing)
  - 1.1.e UPDATE
- 1.2 Troubleshoot these H.323 protocol elements
  - 1.2.b Call set up and tear down
- 1.3 Troubleshoot media establishment

The field of communications has come a very long way since the introduction of the telephone in the 1800s by Alexander Graham Bell. Voice over IP (VoIP) traces its roots back to as early as the 1920s, when the first advancement in reproducing speech electronically and transmitting it over long distances was made. Decades later, in 1974, a significant milestone was achieved when the first voice datagram was transmitted over ARPANET, the precursor to the Internet. The year 1974 also saw another significant milestone in the history of the Internet: the introduction of Transmission Control Protocol (TCP), which would revolutionize the way information was transmitted over the Internet. Experiments carried out in subsequent years adequately demonstrated the need to develop a more flexible protocol for the transmission of real-time traffic classes. This led to the introduction of User Datagram Protocol (UDP), which has gone on to become the default transport layer protocol for real-time applications.

The next big leap in the world of real-time communications occurred in 1995, when an Israeli company by the name of VocalTec pioneered the first widely available Internet phone. At that time, it was possible to make calls between two such phones over the Internet, but speech quality, reliability of connection establishment, and the overall user experience were huge hindrances in preventing VoIP technology from becoming the next big wave in telecommunications.

However, transmitting real-time traffic, like voice and video, over the Internet at a fraction of the cost incurred in circuit-switched networks was such an exciting prospect that equipment manufacturers could not abandon it. The introduction of broadband Internet, with its always-on capability, greatly improved connection reliability, voice quality, and the user experience. This seemed to be the inflection point at which VoIP went mainstream, as corporations realized the immense cost benefits associated with this technology. Consequently, equipment manufacturers invested significant amounts of money and time in developing product lines with an abundance of features and customization options.

Over the next couple years, standards organizations such as the International Telecommunication Union Telecommunication Standardization Sector (ITU-T) and the Internet Engineering Task Force (IETF) took up the task of developing and publishing standards related to VoIP. These standards have become the backbone protocols and enablers on which modern, real-time communications infrastructures operate today.

The two most significant signaling protocols used for real-time communications are Session Initiation Protocol (SIP) and H.323. The ITU-T first standardized the H.323 suite of protocols, which is used to define how multimedia sessions are established. Just a few years later, the IETF began standardizing a competing signaling protocol for VoIP: SIP. It's worth noting that one of the most powerful design elements of SIP was its maximum reuse of existing Internet standards, such as Domain Name System (DNS), Hypertext Transfer Protocol (HTTP), Session Description Protocol (SDP), and Transport Layer Security (TLS). This made it easier for SIP to be deployed and integrated into existing environments while also allowing vendors to reuse these other protocols in their SIP applications.

While H.323 admittedly has a few advantages over SIP, it was the easily consumable HTTP-like text-based approach of SIP and its flexibility, extensibility, and ease of implementation that won out. Thus, SIP has become the de facto signaling protocol for real-time multimedia communications today. For this reason, we have a very brief introduction to H.323 in this chapter but spend more time on the SIP signaling protocol and its use of SDP to describe and negotiate multimedia sessions.

"Do I Know This Already?" Quiz

The "Do I Know This Already?" quiz allows you to assess whether you should read the entire chapter. If you miss no more than one of these self-assessment questions, you might want to move ahead to the "Exam Preparation Tasks" section of the chapter. Table 2-1 lists the major headings in this chapter and the "Do I Know This Already?" quiz questions related to the material in each of those sections to help you assess your knowledge of these specific areas. The answers to the "Do I Know This Already?" quiz appear in Appendix A, "Answers to the 'Do I Know This Already?' Quiz Questions."

Table 2-1 "Do I Know This Already?" Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
Overview of SIP	1–3
Introduction to SDP	4–6
Overview of H.323	7–9

1. Which of the following devices involves colocated UAC and UAS functionality for the forwarding and processing of SIP requests?
   a. SIP proxy
   b. Registrar server
   c. Redirect server
   d. B2BUA
   e. Location server
2. Which of the following messages are server error final responses? (Choose two.)
   a. 404 Not Found
   b. 503 Service Unavailable
   c. 488 Unacceptable Media
   d. 301 Moved Temporarily
   e. 500 Internal Server Error
3. Which headers are required for a SIP INVITE request? (Choose two.)
   a. Call-ID
   b. Expires
   c. Remote-Party-ID
   d. Session-ID
   e. Contact
4. In an early offer call, which two SIP messages carry the SDP message body? (Choose two.)
   a. INVITE
   b. 100 Trying
   c. 200 OK
   d. ACK
   e. BYE
5. Which media codecs require the a=rtpmap SDP attribute due to the use of dynamic payload numbers? (Choose two.)
   a. G711ulaw
   b. G711alaw
   c. OPUS
   d. H.264
   e. G.729

6. Which SIP request can be used to update an existing media session between two user agents? (Choose two.)
   a. INVITE
   b. PRACK
   c. UPDATE
   d. REGISTER
   e. OPTIONS
7. Which H.323 protocol performs the media negotiation during session establishment?
   a. H.225
   b. H.245
   c. H.450
   d. H.264
8. What H.225 TCP port is used to establish non-secure signaling?
   a. 1719
   b. 1720
   c. 1721
   d. 5060
   e. 5061
9. With which of the following is H.245 negotiated before the call connects?
   a. Slow start
   b. Fast start
   c. Early offer
   d. Delayed offer

Foundation Topics

Overview of SIP

Session Initiation Protocol (SIP) forms the backbone of modern real-time communication networks. Over the years, SIP has been enhanced a great deal to include several use cases that make it a very robust multipurpose communication protocol. The following sections provide a brief overview of SIP.

Brief Introduction to and History of SIP

The SIP communications protocol is used for session setup, modification, and teardown. It is an application layer protocol that incorporates many elements of Hypertext Transfer Protocol (HTTP) and Simple Mail Transfer Protocol (SMTP). SIP is modular in design and can work in concert with many other protocols that are required to set up and support communication sessions, including the following:

- Real-Time Transport Protocol (RTP)
- Session Description Protocol (SDP)
- Resource Reservation Protocol (RSVP)

SIP was originally designed by Mark Handley, Henning Schulzrinne, Eve Schooler, and Jonathan Rosenberg in 1996, and it was standardized in 1999 as RFC 2543. The version of SIP standardized in RFC 2543 was 1.0. At the time of this writing, the current SIP version is 2.0, standardized as RFC 3261.

SIP Operation

SIP works on the request/response framework and mirrors a model similar to HTTP, where there is a client/server exchange. A node that generates the request is called a **user agent client (UAC)**, and a node that processes the request and sends out at least one response is called a **user agent server (UAS)**. The concepts of a SIP transaction and a SIP dialog characterize the interaction between the UACs and UASs. A SIP transaction consists of a single request and all responses to that request, which may include zero or more provisional responses (1XX) and one or more final responses (2XX, 3XX, 4XX, 5XX, 6XX). A SIP dialog is a peer-to-peer relationship between user agents that exists for some time.

A single SIP dialog can include multiple SIP transactions. A transaction consists of a single request and the response(s) to that request. Figure 2-1 shows a few of these messages and an example of the SIP dialog with multiple transactions. The first transaction in this figure, known as the INVITE transaction, forms the SIP three-way handshake observed in many SIP dialogs.

Key Topic

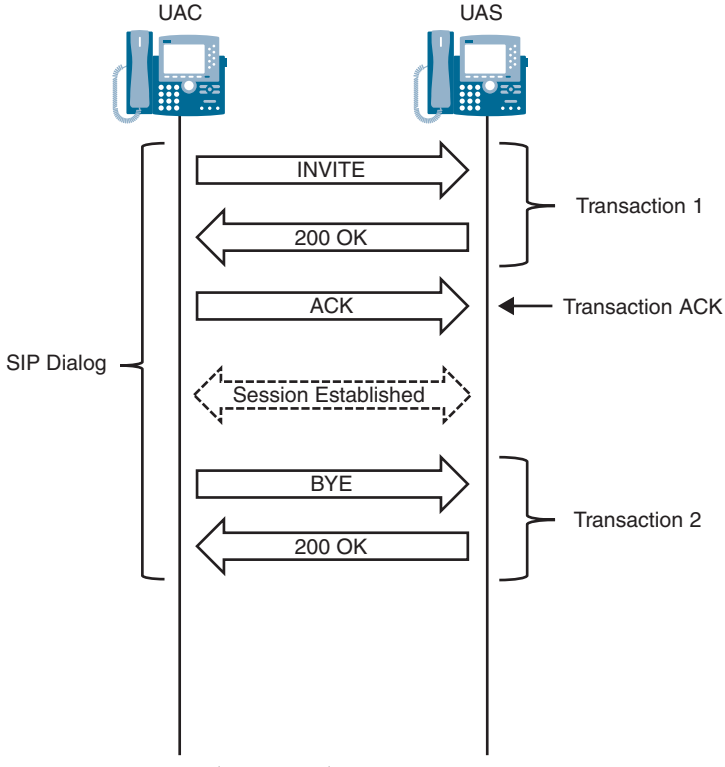


Figure 2-1 *Sample SIP Dialog with Two Transactions*

The following events occur in Transaction 1 in this example:

1. The UAC sends an INVITE message to the UAS in an attempt to establish a session.
2. The UAS sends a 200 OK response, which accepts the INVITE message for the session.
3. The UAC needs to acknowledge the 200 OK message for the INVITE transaction, which is done via an ACK request message. (Note that this message is only observed during the INVITE transaction.)

At this point, the session is established, and it continues until one participant decides to end the session. When that occurs, a second transaction is created, consisting of the following events:

1. Session teardown occurs, with a BYE message sent by one participant (in this case, the originating UAC).
2. The UAS accepts the BYE for session teardown and replies by sending a 200 OK.

NOTE 1XX messages, which are optional, are omitted from Figure 2-1. Similarly, the type of session (for example, audio, video) being established is omitted to keep the example simple.

SIP commonly uses TCP or UDP as the transport protocol. For devices such as call agents, voice gateways, SIP proxies, and session border controllers (SBCs) that typically handle several SIP sessions simultaneously, the transport layer protocol is usually UDP. Establishing and maintaining a connection involves significantly more overhead with TCP than with UDP. However, when SIP sessions need to traverse communication links that are prone to errors, such as packet drops, it is better to use TCP as the transport layer protocol. Port number 5060 is typically used for SIP over UDP or TCP.

SIP messages exchanged between UACs and UASs carry a lot of information that could be misused if it fell into the hands of an attacker. For example, a SIP INVITE carries information that could reveal details of the network topology, the nature of the device originating or servicing the request, and details of the media stream(s), such as the IP addresses and port numbers. This is especially problematic when communication sessions span open networks. To prevent such attacks, it is possible to secure SIP signaling by using Transport Layer Security (TLS). The port number used for SIP over TLS is 5061.

Resources on a SIP network are identified by a uniform resource identifier (URI), which takes the following generic format:

`sip:`*`username`*`:`*`password`*`@`*`host`*`:`*`port`*

If the port is not specified, it defaults to 5060. For secure SIP transmission over TLS, an **s** may be added to the end of **sip** to make it **sips**:

`sips:`*`username`*`:`*`password`*`@`*`host`*`:`*`port`*

Note that the **sips** URI is not required when using TLS, and many implementations use SIP over TLS with the **sip** URI. As with a non-secure **sip** URI, if the port is not specified, a default port is used. In this case, however, the default port is 5061 instead of 5060.

SIP devices are referred to as *user agents* and can be devices such as IP phones, call servers, fax servers, gateways, and SBCs. The originator of a SIP request is called a UAC, and a device that processes the request is called a UAS. SIP includes several functional components, and interactions between user agents in real-world scenarios are in most cases more complex than generic client/server transactions. In order to understand the core tenets of SIP operation, you need to understand the following functional components of SIP:

- **SIP proxy:** A SIP proxy is a device that is capable of performing call routing, authentication, authorization, address resolution, loop detection, and load balancing. A SIP proxy can be stateless or stateful; the fundamental difference between the two is whether they are aware of SIP transactions. A SIP transaction consists of a single request and all responses to that request, which may include zero or more provisional responses (1XX) and one or more final responses.

  A stateful proxy becomes aware of the state of a SIP transaction by creating a server transaction, a client transaction, and a response context. By being transaction aware, the proxy is capable of forking requests, retransmitting requests, and generating messages by itself. For example, a stateful SIP proxy can generate a SIP CANCEL message to all entities still processing a forked request after a final response has already been received.

  Stateless proxies, on the other hand, do not maintain transaction state; they transparently forward requests from the client to the server, and they send responses in the reverse direction. Once a request or a response is forwarded to the intended recipient, all details or transaction context of the message is purged. Consequently, stateless proxies cannot fork requests, retransmit requests, or generate messages on their own.

  Proxies do not manipulate SIP message headers such as To, From, Call-ID, and so on. They do, however, include a Via header and a Record-Route header, and they decrement the Max-Forwards header value by one.

- **Redirect server:** A redirect server is a server that provides location services to user agents or proxies by replying to requests with the location or route to the host that ultimately services the request. This is desirable in situations where there is a need to build highly scalable servers that do not participate in a SIP transaction but simply help the proxy or user agent reach the host by sending a single message. Redirect servers reply to requests with a 3XX response in which the Contact header contains the URI of the location to the host.

- **Registrar server:** A registrar server is a server that accepts registration requests from user agents and creates a mapping between their address of record (AOR)—the public identifier of the user agent—and the user agent's location. Subsequently, this mapping between the user agent AOR and location is indexed and stored in a location server. Cisco Unified Communications Manager (Unified CM) and Cisco Unified Communications Manager Express (Cisco Unified CME) act as registrar servers for Cisco IP phones and other enterprise endpoints. Similarly, Cisco Unified Border Element (CUBE) may be required to register with a service provider. While Unified CM is not covered as a registrar server in this book, the topic of configuring Cisco Unified CME as a registrar server for SIP endpoints is covered in Chapter 10, "Unified CME and SRST." Registrar server interactions for a SIP trunk solution with CUBE are covered in Chapter 9, "CUBE Interworking Features."

- **Location server:** A location server contains a mapping between a user agent's AOR and location; it does not need to be by a separate physical server and can be physically and logically colocated with the registrar server.
- **B2BUA: Back-to-back user agents (B2BUAs)** are devices that have both UAC and UAS functionality and are capable of forwarding requests and processing them. SBCs, such as CUBE, are examples of B2BUAs. Unified CM may also act as a B2BUA. SIP trunking with Unified CM is covered in Chapter 5, "Unified CM SIP Trunk Configuration," and B2BUA operations with CUBE are covered in Chapter 9.

SIP Methods

SIP messages are transmitted in plaintext. SIP messages can be requests or responses to requests. Table 2-2 lists SIP requests and describes the purpose of each one.

Key Topic

Table 2-2 SIP Requests

SIP Request	Description
INVITE	A caller sends out this message to request another entity to establish a SIP session.
ACK	This indicates that the client has received a final response to an INVITE request.
OPTIONS	This request queries the server for its capabilities.
BYE	This is used by the UAC to indicate to the server that it wishes to terminate the established SIP session. (Note that this request can be issued by the caller or the callee.)
CANCEL	This is used to cancel a pending request and can be sent only if the server has not replied with a final response.
REGISTER	A client uses this message to register the address listed in the To header field with a SIP registrar server.
PRACK	This provisional acknowledgment is used to ensure that provisional responses are received reliably. (For more details on this message, see Chapter 9.)
SUBSCRIBE	This creates a subscription for important event notification. (For more details about this message, see Chapter 3, "VoIP Protocols: RTP, RTCP, and DTMF Relay.")
NOTIFY	This notifies the subscriber of the occurrence of an event. (For more details about this method, see Chapter 3.)
INFO	This allows the exchange of application-level information among communicating entities. Information is exchanged without affecting the state of the SIP transaction or dialog.
PUBLISH	This publishes an event to the server.
REFER	This instructs the recipient to contact another entity, using the information specified in this request. (For more details about this message, see Chapter 9.)
UPDATE	This modifies the state of the session without changing the state of the dialog. (For more details about this message, see Chapter 9.)
MESSAGE	This transports instant messages using SIP.

Every SIP transaction begins with a request from a UAC to a UAS. The UAS begins processing the request as soon as it is received. The result of this processing depends on the nature of the request, the formatting of the request, the state of the server at the time the request was being serviced, and the general configuration and policies local to the server. In the case of devices such as SIP proxies, B2BUAs, and voice gateways, the result of processing a request could depend on downstream devices.

SIP servers are always required to respond with the results of request processing. SIP responses use the following formatting convention:

- The SIP version number (2.0 is the current SIP version number)
- A three-digit status code (for example, 404)
- A textual description (for example, Not Found)

The three-digit status code is an integer that communicates the outcome of request processing and is used for machine interpretation. The textual description, on the other hand, is for human observers and is useful in call failure debugging and call record interpretation. The first digit of the status code indicates the SIP response class; there are six classes in all (see Table 2-3).

Key Topic

Table 2-3 SIP Response Classes

Class of Response	Meaning	Type of Response
1XX	Informational: The request has been received and is being processed.	Provisional
2XX	Success: The request has been received, understood, and accepted.	Final
3XX	Redirection: Further action needs to be taken to complete the request. For example, the UAC needs to contact another server to process the request.	Final
4XX	Client error: The request contains bad syntax (such as malformed headers) or could not be fulfilled at the server (for example, if the server could not find the number referenced in the requested URI).	Final
5XX	Server error: A server failed to fulfill a valid request.	Final
6XX	Global failure: The request could not be fulfilled at any server.	Final

Breaking Down a SIP Call

Before diving into the details of how a communication session over SIP is established, it is important to get a sense of how the initiator of a request—the UAC—forms the request and how the UAS ultimately processes the request. The following subsections take a detailed look at how SIP requests and responses are created.

Forming a Request

Standards-based SIP requires that a request contain at least the following header fields:

- Request-URI
- Via
- From
- To
- Call-ID
- Max-Forwards
- CSeq

Subsequent sections discuss these header fields in more detail, and subsequent chapters introduce several other header fields used for specific applications. The header fields that appear in a request can vary depending on the type of request (refer to Table 2-4). For example, a SIP INVITE request would require additional header fields in comparison to a SIP REGISTER request.

Example 2-1 illustrates all of the items discussed in this section. This example shows a SIP INVITE sourced from a Cisco 8865 SIP IP phone acting as a UAC. This INVITE is sent to Unified CM as the UAS for the call. Here you can see the mandatory SIP header fields, such Request-URI, along with Via, From, To, Call-ID, Max-Forwards, and CSeq. This example also shows the required SIP INVITE fields, such as Contact, Allow, Supported, and Accept. Finally, it shows the optional headers, such as User-Agent, Session-ID, Expires, and Remote-Party-ID. These headers and their usage are covered after Example 2-1.

Example 2-1 *Sample SIP INVITE from a Cisco 8865 SIP IP Phone*

```
INVITE sip:2001@rtp-cucm.ccnpcollab.lab;user=phone SIP/2.0
Via: SIP/2.0/TCP 14.50.214.109:49850;branch=z9hG4bK43660dee
From: "+14085557890" <sip:+14085557890@rtp-cucm.ccnpcollab.lab>;tag=c4b36abac
To: <sip:2001@rtp-cucm.ccnpcollab.lab>
Call-ID: c4b36aba-ca23000e-14159e9e-1b38e718@14.50.214.109
Session-ID: 735c6c0600105000a000c4b36abaca23;remote=00000000000000000000000000000000
Max-Forwards: 70
CSeq: 101 INVITE
Contact: <sip:+14085557890@14.50.214.109:49850;transport=tcp>
Allow: ACK,BYE,CANCEL,INVITE,NOTIFY,OPTIONS,REFER,REGISTER,UPDATE,SUBSCRIBE,INFO
Supported: replaces,join,sdp-anat,norefersub,resource-priority,extended-refer
Accept: application/sdp
User-Agent: Cisco-CP8865/12.7.1
Expires: 180
Remote-Party-ID: "+14085557890" <sip:+14085557890@rtp-cucm.ccnpcollab.lab>;
party=calling;id-type=subscriber;privacy=off;screen=yes
```

Several mandatory SIP headers appear in all requests:

- **Request-URI:** In general, each resource within a SIP network is identified by a URI, which is expressed either as a SIP URI or a SIPS URI (SIP Secure URI). Specifically, within the scope of a SIP request, the Request URI header identifies the resource that processes the request.

NOTE In real-world networks, there could be several devices between the UAC and UAS, such as call agents, SIP proxies, and SBCs. While a proxy cannot alter the Request-URI, devices such as SBCs and call agents can modify and transform the Request-URI, if required by local policy or configuration.

- **Via:** This header field indicates the transport layer protocol used for exchanging SIP messages and the location to which responses must be sent. For example, the following Via header field specifies UDP as the transport layer protocol and 10.1.1.1:5060 as the address/port pair for responses:

  ```
  Via: SIP/2.0/UDP 10.1.1.1:5060;branch=z9hG4bK2D1F9D1C08
  ```

  Also included in the Via header field is the branch parameter, which serves as an identifier for the SIP transaction created by any request and remains the same from the perspective of the UAC and the UAS. The branch parameter, which is unique across space and time, is valid until the termination of a SIP transaction. Subsequent requests that create new transactions must ensure that they generate new and unique values for this parameter. When a SIP proxy handles a request, such as a SIP INVITE, it inserts a Via header field before forwarding the request to the next hop. The next hop could be another proxy server or the final destination that processes the request. A request traversing proxies has more than one Via header field value.

- **From:** This header field indicates the logical identity of the user agent that initiates the request. The From header field carries the identity of the initiator of the request in the form of a URI. Optionally, this header field can also include the display name of the initiator. For media sessions established over SIP, the display name in the From header field serves as a caller ID assertion. Intermediary devices such as SBCs usually transform the contents of the From header field. This might be required for a myriad of reasons, such as to enable interoperability across different SIP networks, provide topology abstraction, or make identity assertions. The From header field must carry a tag parameter. (The significance of the tag parameter is explained shortly.)

- **To:** This header field usually identifies the logical entity that is supposed to process the request and is populated using a sip URI/SIPS URI or a tel URI. The logical entity identified in the To header field may or may not be the actual UAS that processes the request. In fact, the entity identified in the To header field usually isn't the one to process the request when the request traverses several hops. Dialog-creating requests (such as SIP INVITE) must never carry a tag parameter in the To header field. Instead, the tag parameter is populated by the user agent that processes the request.

- **Call-ID:** This header field uniquely identifies all messages that belong to a SIP dialog; the SIP dialog in turn is uniquely identified by the combination of the Call-ID header, the From tag, and the To tag. A SIP dialog may contain several SIP transactions. The Call-ID header field value is created by the UAC and retained in messages sent by the UAS. The Call-ID must be unique. User agents must ensure that this header field is generated in such a way to ensure that there isn't any overlap with the Call-ID field of another SIP dialog. The Call-ID header field can sometimes carry the IP addressing information or domain name details of the initiating user agent, which might be undesirable in terms of topology abstraction. SBCs overcome this problem by overwriting the Call-ID header field value and preventing any internal network topology information from crossing network boundaries.
- **Max-Forwards:** This header field value limits the number of hops a request can traverse before it reaches its final destination. Every node that receives a request either partially or completely processes a SIP request. Partial processing could include running syntactical checks, adding header fields, or occasionally modifying a request before it is passed on to the next hop. Complete processing of the request, on the other hand, involves sending one or more of the six response classes listed in Table 2-3 after processing the request. Every node that partially processes the request decrements the Max-Forwards header field value by one before sending the request to the next hop. If a request is received at a user agent or a proxy that is not the final destination of the request, an explicit check is run on the value of the Max-Forwards header field. If it is 0, the request is rejected with a 483 Too Many Hops response. If it is a nonzero value, it is forwarded to the next hop.
- **CSeq:** This header field is used to order transactions within a dialog. It is formatted as follows:

  ```
  CSeq: <Sequence-Number> <Method>
  ```

  where <Method> is a SIP request (refer to Table 2-2).

The header fields listed here are required for every type of SIP request. However, additional header fields might also be required, depending on the type of the SIP method (request). For example, a SIP INVITE requires additional header fields accompanying the mandatory header fields to be efficiently processed by the UAS. For SIP INVITE, the following additional header fields are required:

- **Contact:** This header field provides a SIP or SIPS URI at which the user agent can be contacted for subsequent requests. For example, consider an audio call that is already established between a UAC and a UAS. Due to negotiated session policy or application interactions, the UAS might need to send a midsession request to the UAC. To do so, it uses the SIP or SIPS URI indicated in the Contact header field. Note that the usage of the Contact header field is not restricted to INVITE requests only. This header field is also present in responses to the SIP INVITE and other SIP methods, when applicable.

- **Allow:** It is recommended that the Allow header field be present within a SIP INVITE. This header field advertises the different SIP methods that can be invoked on the UAC within the scope of the dialog initiated by the SIP INVITE request. Parsing the Allow header field allows a UAS to understand the types of requests that can be sent to the UAC during the SIP dialog. For example, if the UAS wants to transmit dual-tone multifrequency (DTMF) information using the SIP INFO message, it can do so only if the UAC-advertised support for the SIP INFO message is in the Allow header field of the INVITE. The following is an example of the Allow header field:

  ```
  Allow:INVITE,ACK,CANCEL,BYE,REGISTER,REFER,INFO,SUBSCRIBE,
  NOTIFY,PRACK,UPDATE,OPTIONS
  ```

  Although it is recommended for UACs to include the Allow header field in INVITE requests, there might be instances when this header field isn't included. In such cases, the UAS must not assume that the UAC does not support any method; rather, it must be interpreted as the unwillingness of the UAC to advertise what methods it supports. The UAS can go ahead and send requests that are required to further advance the communication session; however, if the method is unsupported by the UAC, these requests are rejected by using the 405 Method Not Allowed response.

- **Supported:** A UAC might use this header field to enumerate the various extensions to baseline SIP that it supports. A UAS might apply these extensions to baseline SIP when responding to the request. For example, if the UAC includes the timer extension in the Supported header field, it advertises support for SIP session refresh. The UAS might apply this extension to ensure session liveliness, as per the guidelines of RFC 4028. Session timers and session refresh operations are covered in Chapter 9.

- **Accept:** The UAC might include the Accept header field to indicate content types that are acceptable to it in responses to requests or in new requests within the dialog. This header field allows user agents to advertise support for various session description formats.

Although some of the header fields listed here are optional, they are nonetheless universally supported across device types and vendors for initiating communication sessions over SIP. The following header fields might be added in SIP INVITE requests (although their exclusion does not deter the SIP session from proceeding smoothly):

- **Require:** When included in the SIP INVITE, this header field enumerates extensions to baseline SIP using option tags. Each option tag represents a SIP extension that the server must support in order to process the request. If the server cannot support a specific extension, the request is rejected with a 420 Bad Extension response.

- **Expires:** This header field might be added by a UAC to limit the validity of an invitation. Once a communication session is established, this header field value has no bearing on the amount of time for which the session can last. The usage of the Expires header field is method dependent. (As described in Chapters 9 and 10, this header field has a different purpose for the SIP REGISTER method.)

- **Diversion:** This header field is used when a call is diverted from the original called endpoint to another endpoint. You will observe a Diversion header when a call is forwarded in Unified CM by either the Call Forward No Answer, Call Forward Busy, or Call Forward All setting for a line. This header is covered in both Chapters 5 and 8, "CUBE Call Routing and Digit Manipulation."
- **Remote-Party-ID (RPID):** This header field is primarily used for caller identification. The data from this header field often supersedes caller identification information in other headers, such as the Contact header or From header, and Cisco IP phones use this information to display caller ID information, if present. Another header field that effectively achieves the same thing is the P-Asserted-Identity (PAI) field. For more information about how to configure Unified CM to add these header fields, see Chapter 5.

Table 2-4 summarizes the mandatory, method-dependent, and optional headers for a SIP INVITE message. As mentioned previously, different SIP methods have different method-dependent and optional headers.

Table 2-4 Classification of Headers in SIP INVITE Messages

Header Fields	Requirement
Request-URI, Via, From, To, Call-ID, Max-Forwards, CSeq	Mandatory for all SIP messages
Contact, Allow, Supported, Accept	Required for INVITE messages and optional for other methods
Require, Expires, Diversion, Remote-Party-ID, P-Asserted-Identity	Optional for SIP INVITE messages

It is also possible for vendors to include proprietary header fields in SIP INVITE requests. If such a request happens to be processed by a device from the same vendor, any proprietary header field is interpreted to enable additional functionality. For example, Cisco devices such as Unified CM, voice gateways, and CUBE use the Call-Info header field in SIP INVITE messages (and corresponding responses) to advertise support for Cisco's proprietary method of DTMF relay or SIP unsolicited notify. (For more on this, see Chapter 3.) If a nonmandatory proprietary header cannot be understood by a UAS, it is dropped, and processing of the request continues according to RFC 3261.

Forming a Response

On receiving a SIP request, a UAS performs a sequence of checks to determine how to respond to the request. Figure 2-2 illustrates the logic executed on the server to process a request.

The first check executed at the UAS involves whether the SIP method is supported. SIP methods are nothing but requests that require a specific action to be executed at the server. The SIP requests listed in Table 2-2 are all examples of SIP methods. All SIP user agents that initiate and support calls support the SIP INVITE method; however, it is possible for a UAS to not extend support to all the methods listed in Table 2-2. If a UAS does not support a given SIP method, it responds to the corresponding request with a 405 Method Not Allowed response.

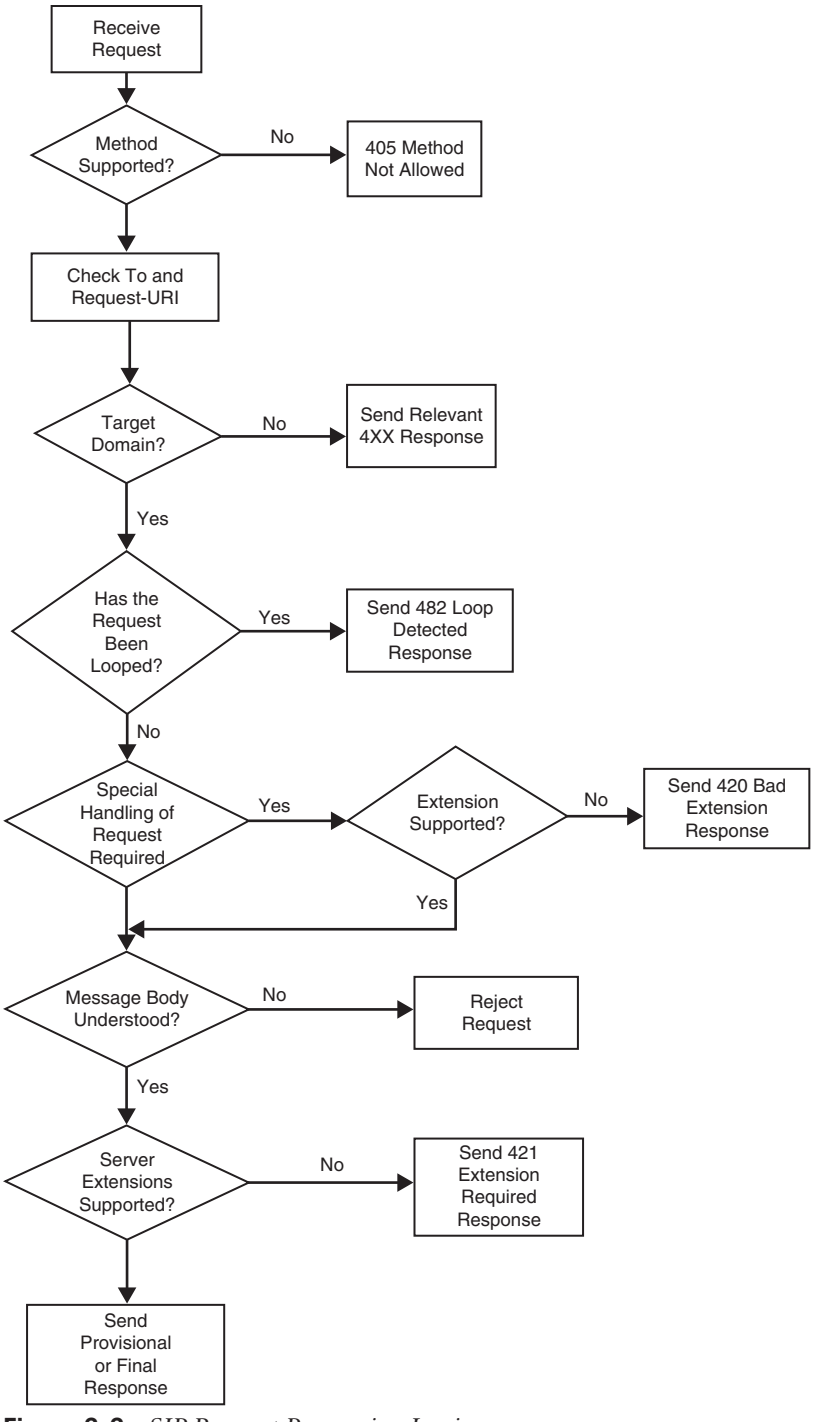


Figure 2-2 *SIP Request Processing Logic*

After inspecting whether the method is supported, the SIP UAS proceeds to check various header fields in the request. The first header fields to be checked are the To and Request URI header fields. These header fields are checked for correct formatting and whether the UAS is indeed the node that is supposed to process the request. If either check fails, the request is rejected via the relevant 4XX class of responses.

The next check that is run is to verify whether the request has been looped—basically, whether the UAS has already processed exactly this request. Looping of requests is quite common in SIP networks, especially when nodes have improperly configured call routing. Call loops result in the UAS rejecting the request with a 482 Loop Detected response.

If a request is determined to be new, the UAS checks whether the client has requested special processing of the request by using the Require header field. This field specifies extensions to baseline SIP that facilitate specific application usage paradigms. These extensions are specified in the form of option tags in the Require header field. For example, the UAC might include the 100rel option tag in an INVITE request to ensure that the server sends provisional responses (101 to 199 responses) reliably. If a UAS does not support an option tag specified by the client, it rejects the request with a 420 Bad Extension response. (For more details on the 100rel option tag, see Chapter 9.)

After the UAS verifies that it supports the extensions required by the client, the next check performed is to determine whether it understands the message body within the request. A request can carry a message body (typically utilizing SDP) that provides additional details about the request. If the UAS cannot interpret a message body within a request, it is rejected.

Finally, in certain cases, a UAS might require the client to support certain extensions to baseline SIP for the request to be successfully processed. For example, if the server requires that provisional responses be sent reliably when a SIP session is being set up, and the UAC does not include the 100rel option tag in the Supported header field, it can reject an INVITE request with a 421 Extension Required response. A UAS should not use the 421 Extension Required response unless it truly cannot process the request using the constructs of baseline SIP.

Once all the checks are executed at the UAS, further processing of the request is strictly method dependent. For example, the same set of rules cannot be used to process an INVITE request and a REGISTER request. While processing a SIP INVITE, the following logic is applied:

- If the INVITE contains an Expires header field, the UAS must send a final response before the expiration interval. If it fails to do so, the request is rejected with a 487 Request Terminated response.
- The received INVITE might be a mid-dialog request (a Re-INVITE). Unless the server undergoes an unexpected restart, it maintains state information for all established dialogs. Mid-dialog requests are usually sent to modify session characteristics or ensure session freshness. For such requests, the processing rules of Section 12.2.2 of RFC 3261 are followed.
- A mid-dialog INVITE request received at the UAS might not match an existing dialog. This could be because of an unexpected server restart, where all dialog contexts are purged, or it might be a result of incorrect request routing by downstream devices.

Whatever the underlying cause, the guidelines of Section 12.2.2 of RFC 3261 are followed to handle such a situation.

If the SIP INVITE is not a mid-dialog request but rather is a dialog-creating request, the UAS is being invited to a communication session. When processing such a request, the UAS can either indicate progress, failure, or success of the request. Alternatively, the request could be redirected:

- **Progress:** If the UAS cannot immediately send a final response to the SIP INVITE message, it can indicate some kind of progress to the request by sending a provisional response between 101 and 199. Commonly used progress indications include the SIP 180 Ringing and 183 Session Progress provisional responses. Provisional responses are classified as early dialog responses and require the UAS to populate the tag parameter of the To header field in the response. A provisional response is usually followed by a 200 OK final response or, in rare cases, a failure response.

- **Failure:** If the UAS is unable to accept the session invitation, a failure response is sent to the client. Processing of the request at the server may fail for a number of reasons. The server could be overloaded, in which case it sends a 503 Service Unavailable response, or it might be that the server could not locate the device specified in the Request URI, in which case it responds with a 404 Not Found response. Whatever the reason, the server responds to the request with an error code that reflects the outcome of processing. The failure response might occasionally carry supplementary information to provide more granular details of the failure and to allow the client to augment the request, if applicable.

  As an example, if the server sends a 503 Service Unavailable response, it can also choose to include a Retry-After header field that provides the client a time after which the request may be retried. If the overloaded condition at the server clears after the specified time interval, the server might respond with a success response.

- **Success:** The session invitation being accepted at the UAS generates a 200 OK response. It is recommended that the 200 response include the Allow, Supported, and Accept header fields. Inclusion of these header fields ensures that the server can advertise any extensions to baseline SIP that it supports without having to be explicitly probed, perhaps via a SIP OPTIONS message.

  Even if the SIP INVITE did not carry a SDP body, the server must include an SDP offer in the 200 OK response. If the INVITE carried an SDP offer, the UAS must include an SDP answer in the 200 OK response.

  200 OK responses to the SIP INVITE must be followed by a SIP ACK method. This ensures that the 200 OK response was delivered reliably.

- **Redirect:** A UAS can respond to INVITE requests with a redirect response (3XX). On receiving a redirect response, the client is required to execute an additional set of actions to complete the request. This usually entails the client parsing the Contact

header field of the redirect response and sending the INVITE message to one or more URIs included in the Contact header field.

When multimedia communication networks peer over SIP, more often than not, 3XX redirect responses are viewed as attempts at toll fraud attacks. Consequently, on receiving a 3XX response, a user agent may terminate the request completely instead of try to further process the request.

Continuing with the transaction started in Example 2-1, Example 2-2 shows the subsequent 200 OK response to that INVITE message. Many of the same header fields observed in the original INVITE are present in this message. The To and From header fields remain unchanged except for a tag that is postfixed on the To header field. This tag is used to identify the specific UAS in the SIP dialog. All future requests and responses between these two UAs should include this To header tag. The Remote-Party-ID and Contact headers will be updated to reflect the appropriate information of the called endpoint or UA handling the request on behalf of the remote endpoint. The Call-ID header will remain the same as seen in the INVITE message, and the CSeq header for this message will reference the same CSeq header of the INVITE request since this is a response to that specific request. The Allow header and Supported header are included to inform the UAC of the allowed request methods and the supported SIP extensions.

NOTE During the INVITE transaction, this message always contains SDP; however, it has been removed here for the sake of simplicity and brevity.

Example 2-2 *200 OK Response from Unified CM to INVITE Sent by the IP Phone*

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP 14.50.214.109:49850;branch=z9hG4bK43660dee
From: "+14085557890" <sip:+14085557890@rtp-cucm.ccnpcollab.lab>;tag=c4b36abac
To: <sip:2001@rtp-cucm.ccnpcollab.lab>;tag=279932
Call-ID: c4b36aba-ca23000e-14159e9e-1b38e718@14.50.214.109
CSeq: 101 INVITE
Allow: INVITE, OPTIONS, INFO, BYE, CANCEL, ACK, PRACK, UPDATE, REFER, SUBSCRIBE,
NOTIFY
Supported: replaces
Server: Cisco-CUCM12.5
Remote-Party-ID: <sip:2001@172.18.110.91>;party=called;screen=yes;privacy=off
Session-ID: 0f3add4b00105000a000d0ec35ffeafe;remote=735c6c0600105000a000c4b36abaca23
Contact: <sip:2001@172.18.110.91:5060;transport=tcp>
```

Analyzing a Basic SIP Call

A SIP call begins when a UAC invites a UAS to a communication session. In real-world implementations, the two user agents might be separated over several hops. However, for the sake of simplicity, let's assume that the two user agents communicate directly with one another.

When inviting another user agent to a communication session, the UAC might be preconfigured with the exact location of the UAS. If not, its request might be ferried by a SIP proxy server to the intended UAS. When the request is received at the UAS, the UAS allocates the necessary resources to process the request. As the request is being processed, the UAS might be required to send provisional responses. It must send a final response once the request is fully processed to alert the UAC of the outcome.

Based on the results of request processing, the server responds with any one of the six classes of response codes listed in Table 2-3. Depending on the request, the UAS may respond with a provisional response class (1XX) followed by a final response class, or it may respond directly with a final response class. For example, when processing a SIP INVITE, the UAS typically sends a 100 Trying provisional response, followed by a final response. The provisional 100 Trying response alerts the UAC that the request is currently being processed, and a final response is expected shortly. It also serves as a mechanism to deter the UAC from sending subsequent copies of the same SIP INVITE.

Figure 2-3 demonstrates a SIP message exchange for establishing an audio call between Phone A and Phone B. A call agent, like Cisco's Unified Communication Manager (commonly known as Unified CM), is in the middle of the signaling for both phones and aids in the establishment of the communication session.

Key Topic

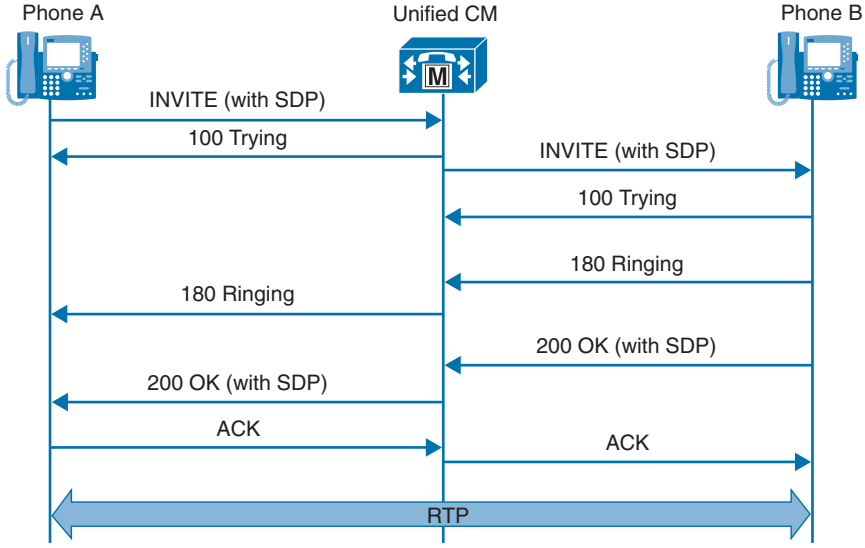


Figure 2-3 *Analyzing a Basic SIP Call*

The SIP call in Figure 2-3 involves the following steps:

Step 1. Phone A initiates a communication session with Phone B by sending a SIP INVITE message to Unified CM. In this scenario, Unified CM functions as both the registrar server for the phones and their UAS for all outbound requests.

Step 2. Included in the SIP INVITE are several pieces of information that enable the transaction/dialog to progress smoothly. These pieces of information include several header field values in the SIP message. The SIP INVITE can be sent in one of two ways:

- With an SDP body
- Without an SDP body

If the SIP INVITE carries an SDP body, the call is classified as an "early offer" call. If an SDP body is not advertised in the SIP INVITE, the call is classified as a "delayed offer" call. As discussed shortly, SDP is used to encode the characteristics of a media session, such as the type of media stream(s) supported (for example, audio, video), the IP addresses and port numbers for the media stream(s), and the set of supported codecs for different media stream types.

Sending an early offer INVITE allows the UAC to enforce characteristics of the session up front by including its supported media stream types, the relevant media formats per media stream, and any SDP-based extensions. With delayed offer INVITE messages, the UAC has to tailor its session characteristics in accordance with the SDP body advertised by the UAS. The example shown in Figure 2-3 illustrates an early offer call.

Step 3. On receiving the SIP INVITE message, Unified CM sends a 100 Trying response to Phone A. The 100 Trying response serves to inform Phone A that the INVITE has been received, and processing is under way. After sending the 100 Trying response, Unified CM examines the Request URI in the received INVITE message and does a database lookup. The database lookup is done to determine the location information (IP address and port) of Phone B. The location information for Phone B is present in Unified CM because it also functions as a registrar server.

Step 4. On obtaining the location information of Phone B, Unified CM operates as a SIP B2BUA, and a SIP INVITE is sent from Unified CM to Phone B. Phone B sends a 100 Trying response to Unified CM.

Step 5. After the request is completely processed at Phone B and it begins ringing, a 180 Ringing message is sent to Unified CM. The 180 Ringing message is then relayed from Unified CM to Phone A. At this stage, an audible ringback tone must be generated at Phone A. The ringback tone might be generated locally on the phone or might be generated by Unified CM. Alternatively, if Phone B wants to stream a custom ringback tone or pre-connect announcement, it sends a 183 Session Progress message with an SDP body. This scenario, defined as "early media," allows Phone A to listen to media packets encapsulating custom ringback tones or pre-connect announcements even before Phone B goes off-hook. For more information about early media and ringback, see Chapter 9.

Step 6. Once Phone B is taken off-hook, a 200 OK response is sent to Unified CM, indicating that the call has been answered. Included in the 200 OK is an SDP body indicating the chosen media stream(s) and media codecs. The 200 OK response is then sent to Phone A. At this stage, the phones can begin to

exchange media packets with one another. The 200 OK response must be followed by a SIP ACK sent end-to-end to indicate that the 200 OK response was reliably received.

At this stage, the SIP dialog is considered complete. You should note that Unified CM is only responsible for setting up the communication session but for most scenarios does not place itself in the path of the media packets.

Step 7. The SIP call terminates when one of the phones transmits a SIP BYE message.

Introduction to SDP

In the earlier days of Internet telephony, the process of setting up a call was very cumbersome and drawn out—very unlike the seamless call setup we experience today. To set up a communication session in the early days, participants had to do the following:

Step 1. The caller would bootstrap an audio or video application at a specific port number and IP address.

Step 2. The caller would inform the callee of the details of the port number and IP address over a PSTN line.

Step 3. The callee would fire up a local audio or video application and inform the caller of the IP address and port number on their end.

While this process was acceptable for the occasional calls made over packet networks for the purpose of research and demonstration, it clearly would not find acceptance if Internet telephony were to scale. Protocols were needed to set up, modify, and tear down communication sessions, and these protocols needed to provide enough information to allow participation within the communication session. SIP, as a call control protocol, is adept at setting up and tearing down communication sessions. However, it does not provide participants any information about the details of the communication session (for example, the media types supported and the IP/port pair for media). As a result, SIP relies on a peer protocol to facilitate advertisement and negotiation of media capabilities.

Session Description Protocol (SDP), originally defined in RFC 2327 (and later updated in RFC 4566), was designed to provide session details (such as the media types, media codec, and IP/port pair for media) and session metadata (such as the purpose of the session and the originator of the session) to participants. SDP is strictly a description protocol and it is leveraged by higher-level protocols such as Session Announcement Protocol (SAP), Session Initiation Protocol (SIP), Media Gateway Control Protocol (MGCP), Real Time Streaming Protocol (RTSP), Multipurpose Internet Mail Extensions (MIME), and Hypertext Transfer Protocol (HTTP).

SDP is completely textual and rigid in terms of formatting. Unlike H.323, it does not use binary encoding, such as ASN.1. This choice was made deliberately so that SDP could be leveraged by several protocols and to ensure that malformed SDP bodies could be easily identified and discarded. The formatting of SDP bodies is mostly in UTF-8. SDP bodies contain a number of textual lines that are each either classified as fields or as attributes. A field is separated from the next one by a carriage return/line feed (CRLF) sequence. The format of each field is as follows:

```
<type>=<value>[CRLF]
```

Attributes are the primary means of extending SDP. Over time, to enable several application use cases and to enable smooth interoperability between communicating entities, many SDP attributes were defined and standardized. (At the time of this writing, there are a few new SDP attributes in the process of being standardized by the IETF.) Attributes can be of two types:

- **Property attributes:** These attributes are in the format a=<flag>. A property attribute conveys a simple Boolean meaning for media or the session.
- **Value attributes:** These attributes are in the format a=<attribute>:<value>.

The primary purpose of an SDP body is to always ensure that the participants are provided sufficient information to join a communication session. Accordingly, SDP bodies are classified into three description levels:

- Session description
- Time description
- Media description

The session description consists of a number of fields and optional attributes that provide details around the session, such as the name of the session, the originator of the session, and bandwidth constraints for the session. The session description can optionally contain attributes as well.

Communication sessions can either be unbounded or bounded in time. SDP time descriptions specify when communication sessions are active by using the timing (t=) field. The timing field has the following format:

```
t=<start-time> <stop-time>
```

This field is self-explanatory: *start-time* and *stop-time* simply encode the time when the session starts and ends, respectively. *start-time* and *stop-time* are expressed in decimal representations of Network Time Protocol (NTP) time values in seconds since 1900. The encoding of the *start-time* and *stop-time* determines whether the communication session is bounded, unbounded, or permanent. A bounded session has an explicit *start-time* and *stop-time*. An unbounded session does not have a *stop-time*, whereas a permanent session does not have a *start-time* or *stop-time*. The encoding of the timing field is useful for multicast communication sessions. For unicast sessions, the timing field must be encoded to specify a permanent session (t=0 0).

The media description section of SDP bodies provides sufficient detail about the media and transport characteristics of the communication session. Participants use this information to join a multicast session or negotiate common capabilities for unicast sessions. The media description section includes the following information:

- The media types (for example, audio, video, application, image)
- The transport protocol (for example, RTP)
- The media formats for different media types (for example, G.711, H.264)
- Optionally, the IP address and port pair for media

Fields and attributes in SDP bodies can be either mandatory or optional. In either case, they must follow the rigid ordering structure shown in Table 2-5.

Key Topic

Table 2-5 Fields and Attributes in SDP Bodies

Field/Attribute	Description	Mandatory or Optional?
Session Description		
v=	Protocol version	Mandatory
o=	Originator and session identifier	Mandatory
s=	Session name	Mandatory
i=	Session information	Optional
u=	URI of description	Optional
e=	Email address	Optional
p=	Phone number	Optional
c=	Connection information; not required if included in all media	Optional
b=	Zero or more bandwidth information lines	Optional
z=	Time zone adjustments	Optional
k=	Encryption key	Optional
a=	Zero or more session attribute lines	Optional
Time Description		
t=	Time the session is active	Mandatory
r=	Zero or more repeat times	Optional
Media Description (If Present)		
m=	Media name and transport address	Mandatory
i=	Media title	Optional
c=	Connection information; optional if included at the session level	Optional
b=	Zero or more bandwidth information lines	Optional
k=	Encryption key	Optional
a=	Zero or more media attribute lines	Optional

SDP fields and attributes can appear at two levels:

- Session level
- Media level

The session-level section of SDP bodies provides default values for various fields that are to be used and interpreted. For example, if a user agent wants to use the same media connection IP address for all media streams within the session, it can encode an SDP body with a session-level description of media connection information. However, if further granularity is required on a per-media-stream basis, the user agent can encode an SDP body with one or several media-level descriptions. Example 2-3 is a snippet of an SDP body carried within a SIP message. (The actual SIP message is omitted from this example for brevity.)

Example 2-3 *SDP Body Carried Within a SIP Message*

```
v=0
o=CiscoSystemsSIP-GW-UserAgent 1597 5834 IN IP4 10.94.64.12
s=SIP Call
c=IN IP4 10.1.1.1
t=0 0
a=recvonly
m=audio 16590 RTP/AVP 8 101
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=ptime:20
m=video 51372 RTP/AVP 126
a=rtpmap:126 H264/90000
```

The session-level description starts with the v= line and continues until the first media-level section. Every media-level section is identified by an m= line and continues until the next m= line or until the end of the SDP body. As shown in Example 2-3, the media connection IP address (c=IN IP4 10.1.1.1) has only a session-level description, and it spans the audio and video stream. In addition to having a media connection information field, the direction attribute (a=recvonly) is specified for both the audio and video media streams. Session-level descriptions serve as default values to be interpreted and used if no corresponding media-level description(s) is available.

Example 2-4 is a snippet of an SDP body where the direction attribute has a session-level description and a media-level description. You should be aware that certain SDP fields and attributes can be present concurrently at different levels of the SDP body. When this occurs, the media-level field or attribute overrides the session-level field or attribute. So, in the case of the direction attribute appearing twice in Example 2-4, the media-level description of the direction attribute is given higher precedence.

Example 2-4 *SDP Body with Session- and Media-Level Definitions for the Direction Attribute*

```
v=0
o=CiscoSystemsSIP-GW-UserAgent 1597 5834 IN IP4 10.94.64.12
s=SIP Call
c=IN IP4 10.1.1.1
t=0 0
a=recvonly
m=audio 16590 RTP/AVP 8 101
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=sendrecv
a=ptime:20
```

The Offer/Answer Framework

SDP was originally conceived as a way to describe multicast sessions over Mbone (short for *multicast backbone*). SDP scales really well for multicast, as there is a unified view of the session for all participants. For example, for multicast communication sessions, each participant requires a single media address and port to join a communication session. While SDP has the capability of describing unicast communication sessions, it is a slightly more challenging proposition than describing multicast sessions. For a unicast session between two participants, each participant has a localized view of the session; each participant has its own media IP address and port pair, its own set of supported media types, and its own set of supported codecs per media type. To obtain a complete view of a unicast session, the participants must exchange information elements and agree on a common set of parameters. The SDP offer/answer model, defined in RFC 3264, provides such a framework for information exchange and parameter negotiation. To get a better understanding of the offer/answer framework, it is important to understand certain terms that are frequently referenced in subsequent sections:

- **Agent:** An entity involved in an offer/answer exchange
- **Answerer:** An agent that receives a session description that describes aspects of a plausible media communication session and responds with its own session description
- **Answer:** An SDP message sent from an answerer to an offerer
- **Offerer:** An agent that generates a session description to create or modify a session
- **Offer:** An SDP message sent by an offerer
- **Media Stream:** A single media instance in a communication session

Operation of the Offer/Answer Framework

The offer/answer exchange requires the existence of a stateful, higher-level protocol such as SIP that is capable of exchanging SDP bodies during call setup and/or modification. The protocol has to be stateful to maintain context around the exchange between an offerer and an answerer, as there may be several SDP exchanges during the course of a call. It is important for the higher-level protocol to accurately map requests and responses.

Generating the SDP Offer and Answer

The SDP offer/answer model begins with one of the user agents constructing an SDP body according to the guidelines of RFC 4566. You should realize that the initiator of a communication session (the user agent that sends the SIP INVITE) need not always be the one constructing the SDP offer. For example, for SIP delayed offer calls, the user agent being invited to a communication session is the one that constructs the SDP offer. Figure 2-4 shows the SIP three-way handshake for a delayed offer call versus the same handshake for an early offer call.

Key Topic

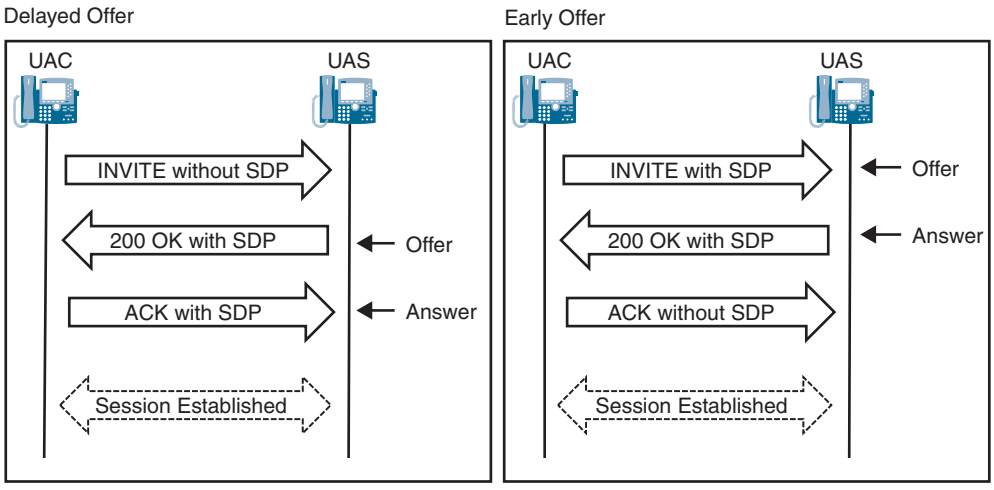


Figure 2-4 *SIP Delayed Offer Versus SIP Early Offer*

Regardless of which user agent constructs the offer, the SDP body must consist of a session description, a time description, and a media description section. This strict encoding format ensures that the peer user agent is provided sufficient information to participate in the communication session. The session description section contains all the mandatory fields (for example, v, o, s) as well as optional attribute values. For unicast sessions, the time description section must contain a timing field to indicate a permanent session (t=0 0). The media description section of the SDP offer can contain several media lines (m=), such that each media line corresponds to different media types or they correspond to the same media type or a combination of the two.

Each media line in the SDP body must encode sufficient information about the media stream to convey the following:

- The media type of the stream (for example, audio, video, image)
- The transport port and IP address of the media stream
- The list of media formats per media stream

The format of any media line within an SDP body is as follows:

```
m=<media> <port> <proto> <fmt list>
```

The <media> subfield indicates the media type, such as audio, video, image, and so on. The possible set of media types that can be advertised in SDP bodies is maintained in the Media Type registry of the Internet Assigned Numbers Authority.

The <port> subfield is used to advertise the port number on which media reception is expected. It is a common misconception that the port number signifies the port number from which media is sourced. Although most implementations utilize symmetric RTP, which does source RTP from the same port advertised by SDP for RTP reception, some implementations may utilize asymmetric RTP, which may use another source port. For media transport protocols such as RTP, the peer protocol Real-Time Transport Control Protocol (RTCP)

allows participants to provide real-time media reception quality feedback. By default, RTCP is exchanged on the next higher port number following the RTP port number. If for some reason an application does not want to exchange RTCP on the next higher port number following RTP, it can explicitly indicate this by using the a=rtcp attribute. Example 2-5 demonstrates the use of the a=rtcp attribute in an SDP body.

Example 2-5 *SDP Body Demonstrating the Use of the a=rtcp Attribute*

```
v=0
o=CiscoSystemsSIP-GW-UserAgent 1597 5834 IN IP4 10.94.64.12
s=SIP Call
c=IN IP4 10.1.1.1
t=0 0
a=recvonly
m=audio 16590 RTP/AVP 8 101
a=rtcp:53020
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=sendrecv
a=ptime:20
```

The <port> subfield is useful only when interpreted and used in conjunction with the connection data (c=) field. Without the connection data field, the remote user agent is only aware of a port number and has no information about the remote IP address. The connection data field can be scoped to include a session-level or media-level definition.

The <proto> subfield identifies the transport protocol for media. For media encapsulated over RTP, this subfield is set to RTP/AVP or, optionally, to RTP/SAVP for Secure RTP (SRTP). AVP stands for audio/video profile, and SAVP stands for secure audio/video profile.

The <fmt list> subfield specifies the media formats supported by the user agent generating the SDP body. The encoding of this subfield depends on the value of the <proto> subfield, which is either set to RTP/AVP or RTP/SAVP and includes a list of payload numbers (or sometimes only one payload number). For applications to discern the media format to which a given payload number corresponds, there is a list of payload number-to-media format mappings defined in the RTP audio/video profile. Table 2-6 lists a selection of these mappings for common audio codecs.

NOTE A comprehensive list of the mappings of all payload numbers to media formats is maintained in an Internet Assigned Numbers Authority registry at https://www.iana.org/assignments/rtp-parameters/rtp-parameters.xhtml.

Table 2-6 Mapping Between Payload Numbers and Media Formats for Common Audio Codecs

Payload Type	Encoding Name	Clock Rate (Hz)
0	PCMU	8000
4	G723	8000
8	PCMA	8000
9	G722	8000
15	G728	8000
18	G729	8000

In the case of dynamic payload numbers (payload numbers between 96 and 127), there has to be an explicit mapping specified in the SDP body, using the a=rtpmap attribute. While it is not required to use the a=rtpmap attribute for static assignments already specified in the RTP audio/video profile, it seems to be the preferred formatting choice for most vendors.

To better understand this concept, see the sample SDP body provided in Example 2-6. The media line (m=) lists three static payload numbers: 0, 8, and 18. For a user agent that receives this SDP body, the interpretation of the static payload numbers 0, 8, and 18 is provided by the RTP audio/video profile and translates to PCMU, PCMA, and G729, respectively (refer to Table 2-6). Providing a mapping between these static payload numbers and their corresponding media formats via the a=rtpmap attribute is a redundant but nonetheless well adopted practice. For dynamic payload numbers, such as the OPUS codec utilizing payload type 114, the a=rtpmap attribute is required to explicitly provide a binding to the media format.

Example 2-6 *SDP Body Demonstrating the Use of the a=rtpmap Attribute*

```
v=0
o=CiscoSystemsCCM-SIP 2828060 1 IN IP4 10.1.1.1
s=SIP Call
c=IN IP4 10.1.1.1
b=TIAS:64000
b=AS:64
t=0 0
m=audio 17236 RTP/AVP 0 8 18 114
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:114 opus/48000/2
a=fmtp:114  maxplaybackrate=16000;sprop-maxcapturerate=16000;maxaveragebitrate=64000
;stereo=0;sprop-stereo=0;usedtx=0
```

The a=fmtp attribute shown in Example 2-6 is used to encode media format–specific parameters. For example, when using the OPUS codec, many attributes can be utilized. One example defined here is the maximum average bitrate for the session via the maxaveragebitrate attribute. As the example shows, many attributes can be applied to a given media format on a single line by using the semicolon (;) as a separator. These attributes vary per media format, so refer to the applicable media format standard for a=fmtp usage.

Media lines have their interpretations tightly coupled with the SDP direction attribute. A direction attribute can have a session-level scope or a media-level scope. For unicast streams, the offerer can specify the directionality of a media stream by using the SDP direction attribute. Accordingly, a stream can be marked as sendonly, recvonly, inactive, or sendrecv. Table 2-7 summarizes the meaning of each Direction Attribute.

Key Topic

Table 2-7 SDP Direction Attributes Description

Direction Attribute	Direction of Media
sendonly	The sender wishes to only send media to its peer.
recvonly	The sender wishes to only receive media from its peer.
sendrecv	The sender wishes to send and receive media.
inactive	The sender wishes to set up the session but not transmit or receive media.

When the direction attribute has a sendrecv or recvonly value, it signifies the IP address and port number on which the sender would expect to receive media (RTP) from its peer. If the direction attribute is marked as sendonly, it indirectly signifies the IP address and port on which the sender (of the SDP) expects to receive RTCP but not RTP.

As mentioned previously, the IP address and port listed in the SDP offer does not signify the source address and port for RTP packets. Instead, it signifies the address and port on which the offerer expects to receive media.

If a user agent sets the direction attribute to inactive, it means that the user agent wants to simply establish a communication session without transmitting or receiving media. However, at a later time, the user agent can initiate a new SDP offer/answer exchange to update the direction attribute. Regardless of the value of the direction attribute, there is a continuous passage of RTCP traffic between communicating entities. While constructing an SDP body, if the user agent does not specify an explicit value for the direction attribute, it always defaults to sendrecv.

As mentioned earlier, the user agent constructing the SDP offer can include one or more media lines such that the media lines can correspond to the same media type, different media types, or a combination of the two. Conventionally, the offerer must use a valid, non-zero port number for each media line within the offer. This is because the use of port zero for a media line(s) within the offer has no useful semantics.

On receiving the offer, the answerer must construct an SDP body following the guidelines of RFC 4566: It must include a session description, a time description, and a media description. Even if there is absolute parity between the offer and the answer in terms of the media streams and media formats per stream, it is reasonable to assume that the answer will differ from the offer on certain aspects such as the IP address and port pair for media, support for SDP extensions, and so on. In such instances, the origin line (o=) of the answer must be different from that of the offer. The timing field in the answer must mirror the timing field in the offer. With regard to the media description, the constructed answer must follow several rules that are discussed in more detail in the following paragraphs.

While constructing the answer, the answerer must generate a response to each media line listed in the offer, and the number of media lines in the offer and answer must always be the same. If a given media type in the offer is not supported by the answerer, the answerer must

reject the corresponding media line by setting the port number to zero. If an answerer rejects a media stream, there is no RTCP traffic exchanged for that media stream. Example 2-7 demonstrates an offer/answer exchange where the video media type is rejected by the answerer.

Example 2-7 *SDP Offer/Answer Exchange for Disabling Video*

```
[Offer]
v=0
o=Cisco-SIPUA 23226 0 IN IP4 14.50.214.109
s=SIP Call
c=IN IP4 14.50.214.109
t=0 0
m=audio 49170 RTP/AVP 0 8 97
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:97 iLBC/8000
m=video 51372 RTP/AVP 126 97
c=IN IP4 14.50.214.109
b=TIAS:4000000
a=rtpmap:126 H264/90000
a=fmtp:126 profile-level-id=428016;packetization-mode=1;level-asymmetry-allowed=1;
max-mbps=108000;max-fs=3600;max-rcmd-nalu-size=256000
a=imageattr:* recv [x=800,y=480,q=0.60] [x=1280,y=720,q=0.50]
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=428016;packetization-mode=0;level-asymmetry-allowed=1;
max-mbps=108000;max-fs=3600;max-rcmd-nalu-size=256000
a=imageattr:* recv [x=800,y=480,q=0.60] [x=1280,y=720,q=0.50]


[Answer]


v=0
o=CiscoSystemsCCM-SIP 279932 1 IN IP4 172.18.110.91
s=SIP Call
c=IN IP4 14.50.214.107
b=AS:384
t=0 0
m=audio 49172 RTP/AVP 8
a=rtpmap:8 PCMA/8000
m=video 0 RTP/AVP 126
a=rtpmap:126 H264/90000
```

NOTE If there are no media formats in common for all streams, the entire offer session is rejected.

As discussed previously, an offerer can set the direction attribute (at the session level or media level) to sendrecv, sendonly, recvonly, or inactive. Table 2-8 highlights the different ways in which the direction attribute can be set in an answer for unicast media sessions.

Key Topic

Table 2-8 Different Ways of Setting the Direction Attribute in an Answer

Direction Attribute in Offer	Direction Attribute in an Answer
sendonly	recvonly/inactive
recvonly	sendonly/inactive
sendrecv	sendrecv/sendonly/recvonly/inactive
inactive	inactive

For streams that are marked as recvonly in the answer, the answer must contain at least one media format that was listed in the offer. In addition, the answerer may include media formats not listed in the offer that the answerer is willing to receive. This is useful in scenarios where the offerer proceeds to modify the communication session at a later stage and includes an updated media format list.

For streams that the answerer marked as sendonly, the answer must contain at least one media format that was listed in the offer. For streams marked as sendrecv in the answer, the answer must list at least one media format that it is willing to use for both sending and receiving media. In such a situation, the answer might also list media formats that were not a part of the offer. Again, this is useful in scenarios where the offerer proceeds to modify the communication session at a later stage and includes an updated media format list. For streams marked as inactive in the answer, the media format list in the answer mirrors that in the offer.

NOTE Media formats in the offer and answer are always listed in decreasing order of precedence, from left to right.

It is required for the answerer to use the a=rtpmap attribute for each media format to provide a payload number to the media format binding—regardless of whether the answer contains static or dynamic payload numbers. If a media format in the offer is described using the a=fmtp attribute, and that media format is echoed in the answer, the answerer must ensure that the same fmtp parameters are listed.

NOTE The offer and answer can optionally include bandwidth and packetization interval attributes. For more on packetization intervals, see Chapter 3.

Media lines marked as sendonly and recvonly by the offerer have a reverse interpretation when accepted by the answerer. For example, consider an offer where the audio media line is marked as sendonly. When accepted by the answerer, the same audio media line has to be marked as recvonly. This ensures that the offer/answer exchange concludes with both user agents converging on a unified view of the communication session. In this case, the offerer only transmits media packets, while the answerer only receives media packets. Of course, this assumes that the answerer does not set the stream as inactive.

Modifying a Session

During the course of a communication session, it is not uncommon for application interactions to require modification of session characteristics. These modifications could include changing the media formats, changing the value of the direction attribute, adding new media streams, and removing existing media streams, for instance. Some examples of scenarios where modifications occur are during supplementary service actions such as call hold, resume, transfer, and even conferencing. Nearly all aspects of a communication session can be modified. To effect a change or modification of session characteristics, the two user agents must engage in a new SDP offer/answer exchange. The high-level flow of modifying a communication session is depicted in Figure 2-5.

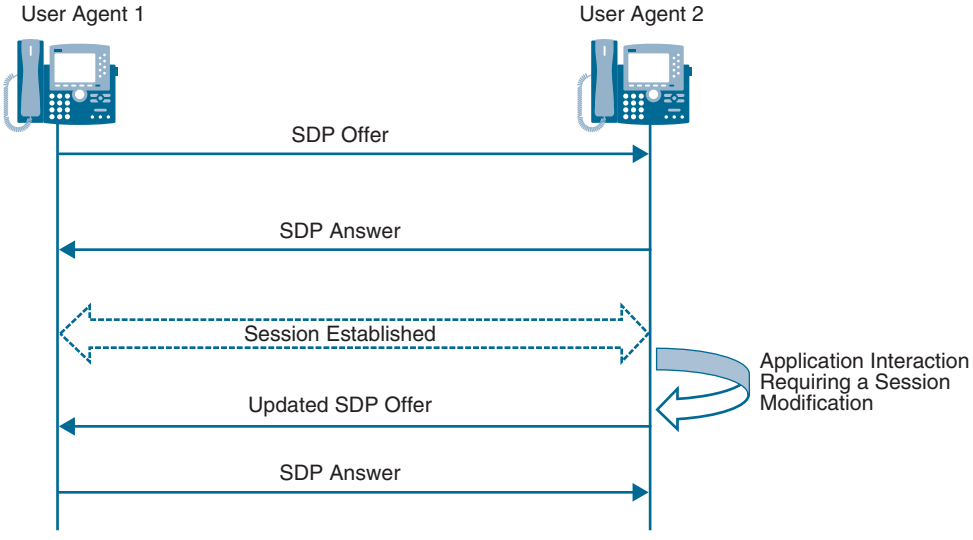


Figure 2-5 *Modifying a Communication Session by Using the SDP Offer/Answer Exchange*

A user agent attempting to modify a communication session first constructs an updated SDP body whose content reflects the modifications required. These modifications can be simple or complex. Regardless of the degree of modification reflected in the SDP body, the user agent must increment the version number of the origin field (o=) by one. Example 2-6 highlights this concept.

The original SDP body in Example 2-8 contains the version number 5834. Sometime during the course of the communication session, the user agent requires a change to the list of media formats and accordingly proceeds to construct and transmit an updated SDP body. The updated SDP offer has its version number incremented by one and a modified media format list in the audio media line. This example also includes two very important SIP headers, which are included in the SIP message to describe and identify the contents of the SIP message body. For an SDP body, the Content-Type value application/sdp is used. Similarly, the Content-Length header field provides the total length of the SDP message body. If no message body is present in the SIP message, the Content-Length header field is set to 0. For the sake of brevity, the SIP header in Example 2-8 only displays the Content-Type and Content-Length SIP header fields.

Example 2-8 *Incrementing the SDP Originator Version Number*

```
[Original SIP+SDP]

Content-Type: application/sdp
Content-Length: 283

v=0
o=CiscoSystemsSIP-GW-UserAgent 1597 5834 IN IP4 10.94.64.12
s=SIP Call
c=IN IP4 10.1.1.1
t=0 0
a=recvonly
m=audio 16590 RTP/AVP 8 101
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=ptime:20
m=video 51372 RTP/AVP 126
a=rtpmap:126 H264/90000


[Modified SIP+SDP]

Content-Type: application/sdp
Content-Length: 227

v=0
o=CiscoSystemsSIP-GW-UserAgent 1597 5835 IN IP4 10.94.64.12
s=SIP Call
c=IN IP4 10.1.1.1
t=0 0
a=recvonly
m=audio 16590 RTP/AVP 0
a=rtpmap:0 PCMA/8000
a=ptime:20
m=video 51372 RTP/AVP 126
a=rtpmap:126 H264/90000
```

It is possible for a user agent to initiate a new SDP offer/answer exchange without changing the contents of the SDP body. While this exchange doesn't result in the modification of session characteristics, it could be used for reasons such as determining session freshness. If a new SDP body is identical to the previous SDP body, the version number must remain the same.

While generating an updated offer, the user agent must ensure that the number of media lines (m=) equals the number of media lines in the previous SDP body. In other words, if the previous SDP body had N media lines, the updated SDP body must contain at least N media lines. It is possible for the updated SDP body to contain more than N media lines since this is required when adding a new media line.

If SIP is the signaling protocol used to establish a media session (with SDP message body for media description), an existing session can be modified using either a re-INVITE or an UPDATE SIP message. Figure 2-6 illustrates both of these scenarios.

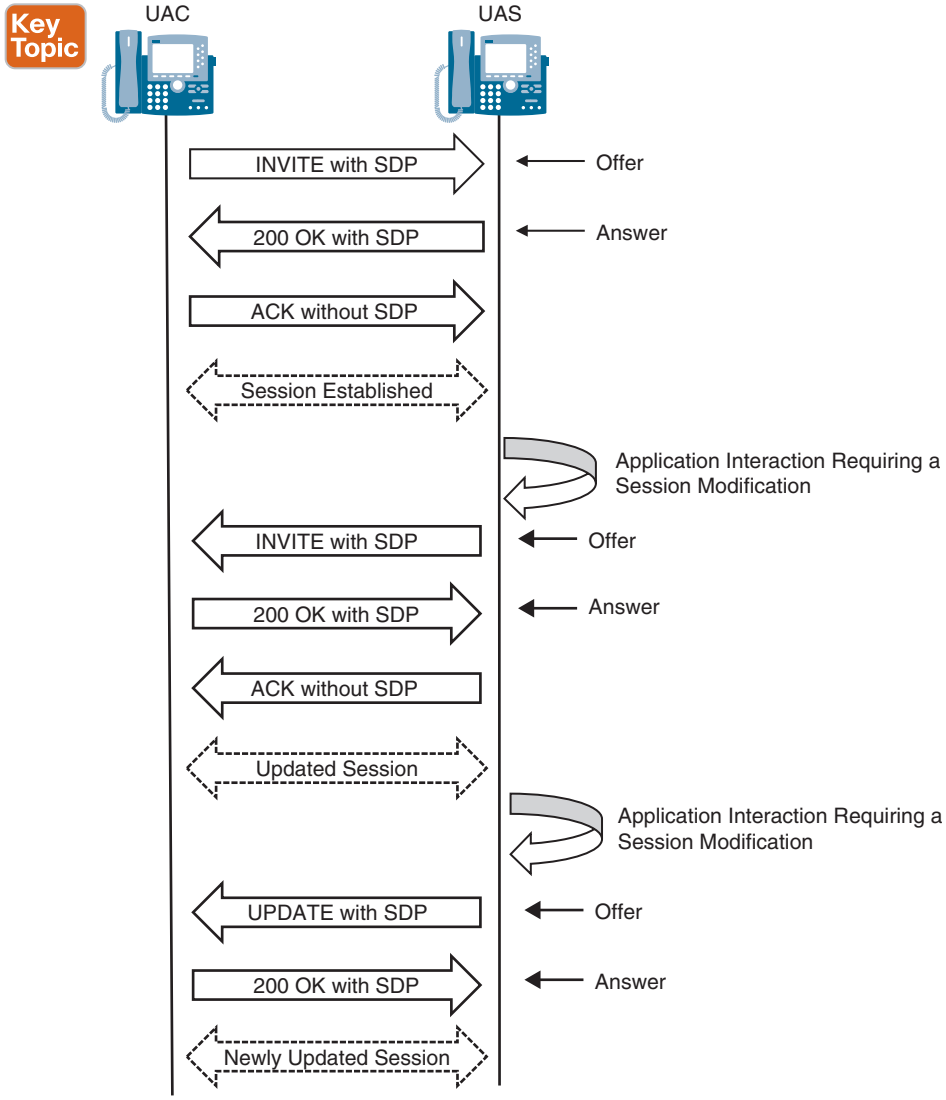


Figure 2-6 *SIP re-INVITE and UPDATE Session Modifications*

The following steps occur in the example shown in Figure 2-6:

Step 1. The initial session is established using an early offer signaling exchange.

Step 2. An application interaction occurs and requires a change in media. This then triggers a mid-call re-INVITE, with a new offer modifying the original session.

Step 3. The UAS accepts the session modification and sends a 200 OK with an SDP body.

Step 4. An ACK to the 200 OK confirms the transaction and finishes the handshake.

Step 5. The session continues with the newly established session parameters until another application interaction occurs. At this point, the session is modified again—this time using an UPDATE SIP message with an SDP body containing an offer for the new session.

Step 6. The remote party accepts this UPDATE with an SDP body via a 200 OK, and an SDP body confirms the session modification.

Step 7. The session continues with the newly updated session parameters.

NOTE This process of modifying the session through re-INVITE or UPDATE SIP message can continue as many times as needed by either user agent involved in the session. Also, note that the UPDATE transaction does not contain an ACK SIP message because the ACK is exclusive to the INVITE/re-INVITE transaction.

Adding a Media Stream

It is possible to add new media streams to a session by appending the appropriate media lines to an existing SDP body. For example, if an audio-only communication session is established between two participants and one of the participants wants to escalate the session to include video, that participant appends a video media line (m=video) to the existing SDP body and sends an updated offer. User agents that want to add a media stream must always append media lines to existing ones. This ordering ensures that the peer user agent is able to gauge any new media line additions. For example, Example 2-9 shows a session established between a video-capable phone and a phone that does not support video. The original video-capable IP phone is then transferred to another video-capable phone, and a new video session is added to the existing media sessions.

Example 2-9 *Adding a Media Stream*

```
[Initial Call (SIP+SDP) - Offer]

Content-Type: application/sdp
Content-Length: 707

v=0
o=CiscoSystemsCCM-SIP 280365 1 IN IP4 172.18.110.91
s=SIP Call
c=IN IP4 14.50.214.107
b=TIAS:64000
```

```
b=AS:80
t=0 0
m=audio 23500 RTP/AVP 114 9 124 113 115 0 8 116 18 101
b=TIAS:64000
a=rtpmap:114 opus/48000/2
a=fmtp:114  maxplaybackrate=16000;sprop-maxcapturerate=16000;maxaveragebitrate=64000;
  stereo=0;sprop-stereo=0;usedtx=0
a=rtpmap:9 G722/8000
a=rtpmap:124 iSAC/16000
a=rtpmap:113 AMR-WB/16000
a=fmtp:113 mode-change-capability=2
a=rtpmap:115 AMR-WB/16000
a=fmtp:115 octet-align=1;mode-change-capability=2
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:116 iLBC/8000
a=maxptime:20
a=fmtp:116 mode=20
a=rtpmap:18 G729/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
```

[Initial Call (SIP+SDP) - Answer]

```
Content-Type: application/sdp
Content-Length: 416

v=0
o=CiscoSystemsCCM-SIP 405281 1 IN IP4 172.18.110.61
s=SIP Call
c=IN IP4 14.50.214.106
b=TIAS:64000
b=AS:80
t=0 0
m=audio 21040 RTP/AVP 114 101
b=TIAS:64000
a=rtpmap:114 opus/48000/2
a=fmtp:114  maxplaybackrate=16000;sprop-maxcapturerate=16000;maxaveragebitrate=64000;
  stereo=0;sprop-stereo=0;usedtx=0
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=trafficclass:conversational.audio.aq:admitted
```

```
[Video Escalation (SIP+SDP) - New Offer]

Content-Type: application/sdp
Content-Length: 1499

v=0
o=CiscoSystemsCCM-SIP 405281 5 IN IP4 172.18.110.61
s=SIP Call
c=IN IP4 14.50.214.106
b=TIAS:384000
b=AS:384
t=0 0
m=audio 21040 RTP/AVP 114 9 124 113 115 0 8 116 18 101
b=TIAS:64000
a=rtpmap:114 opus/48000/2
a=fmtp:114  maxplaybackrate=16000;sprop-maxcapturerate=16000;maxaveragebitrate=64000;
  stereo=0;sprop-stereo=0;usedtx=0
a=rtpmap:9 G722/8000
a=rtpmap:124 iSAC/16000
a=rtpmap:113 AMR-WB/16000
a=fmtp:113 mode-change-capability=2
a=rtpmap:115 AMR-WB/16000
a=fmtp:115 octet-align=1;mode-change-capability=2
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:116 iLBC/8000
a=maxptime:20
a=fmtp:116 mode=20
a=rtpmap:18 G729/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=trafficclass:conversational.audio.avconf.aq:admitted
m=video 29584 RTP/AVP 100 126 97
b=TIAS:384000
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=640C16;packetization-mode=1;max-mbps=108000;
  max-fs=3600;max-rcmd-nalu-size=256000;level-asymmetry-allowed=1
a=rtpmap:126 H264/90000
a=fmtp:126 profile-level-id=428016;packetization-mode=1;max-mbps=108000;
  max-fs=3600;max-rcmd-nalu-size=256000;level-asymmetry-allowed=1
a=rtpmap:97 H264/90000
a=fmtp:97 profile-level-id=428016;packetization-mode=0;max-mbps=108000;
  max-fs=3600;max-rcmd-nalu-size=256000;level-asymmetry-allowed=1
a=imageattr:* recv [x=800,y=480,q=0.60] [x=1280,y=720,q=0.50]
a=content:main
```

```
a=rtcp-fb:* nack pli
a=rtcp-fb:* ccm fir
a=rtcp-fb:* ccm tmmbr
a=trafficclass:conversational.video.avconf.aq:admitted


[Video Escalation (SIP+SDP) - New Answer]

Content-Type: application/sdp
Content-Length: 830

v=0
o=CiscoSystemsCCM-SIP 280365 5 IN IP4 172.18.110.91
s=SIP Call
c=IN IP4 14.50.214.109
b=TIAS:384000
b=AS:384
t=0 0
m=audio 25106 RTP/AVP 114 101
b=TIAS:64000
a=rtpmap:114 opus/48000/2
a=fmtp:114  maxplaybackrate=16000;sprop-maxcapturerate=16000;maxaveragebitrate=64000;
  stereo=0;sprop-stereo=0;usedtx=0
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=trafficclass:conversational.audio.avconf.aq:admitted
m=video 28130 RTP/AVP 100
b=TIAS:320000
a=rtpmap:100 H264/90000
a=fmtp:100 profile-level-id=640C16;packetization-mode=1;max-mbps=108000;
  max-fs=3600;max-rcmd-nalu-size=256000;level-asymmetry-allowed=1
a=imageattr:* recv [x=800,y=480,q=0.60] [x=1280,y=720,q=0.50]
a=content:main
a=rtcp-fb:* nack pli
a=rtcp-fb:* ccm fir
a=rtcp-fb:* ccm tmmbr
a=trafficclass:conversational.video.avconf.aq:admitted
```

It is also possible to add a media stream to a communication session by activating a previously disabled media stream. Consider a scenario in which a user agent attempts to establish a communication session that includes audio and video media types. If the peer user agent does not support video, it rejects the video media line by setting the port to zero (refer to Example 2-7). During the course of the communication session, either user agent may decide to reuse the video media line slot to introduce a new media stream. The new media stream can have a video media type or any other valid media type.

Removing a Media Stream

A user agent can remove an existing media stream by constructing a new SDP body and setting the media port of the corresponding media stream to zero. When such an SDP body is received by the peer user agent, it is treated as a non-negotiable and explicit indication to disable a given media stream. Therefore, the peer user agent must construct an answer with the port for the media stream in question also set to zero.

Media streams that are deleted by an updated SDP offer/answer exchange cease to exchange RTP or RTCP traffic. Any resources allocated for such media streams can be de-allocated.

NOTE The concept of zeroing out a port may also be referred to as *disabling a media stream*.

Modifying the Address, Port, Transport, or Media Format

As mentioned earlier in this chapter, nearly every aspect of a communication session can be modified. The way in which media streams can be added and removed using the SDP offer/answer exchange is discussed in the previous section. During the course of a communication session, it may happen that a user agent discovers a new interface that is known to be more reliable than the current interface engaged in media transmission and reception. To ensure that the newly discovered, higher-priority interface takes over for media transmission and reception, the user agent has to construct an updated SDP offer in which the connection information field is modified to reflect the new interface identity. In most instances, a modification of the connection information field proceeds with a change in the port number(s) for a media line(s), and this is reflected in the update SDP offer.

NOTE There are several other scenarios that require modification of the media IP address and port. These include, but are not limited to, media redirection from one endpoint to another, call hold, and call resume.

Even after updating the connection information and port, a user agent must be prepared to receive media on the old IP address/port pair for a reasonable amount of time. This is because the peer user agent has to accept the updated SDP offer, proceed to process the changes, and then program its internal software subsystems accordingly. You should also be aware that it is possible for an answerer to update its own IP address/port pair in the answer to an updated offer.

When setting up a communication session, the participants converge on a media format by using the SDP information that is exchanged. In addition, it is perfectly acceptable for a user agent to attempt to change the media format midsession. The way in which a user agent changes the media format midsession is achieved by first constructing an updated SDP offer such that the media line(s) contains a completely new set of media formats (not present in the previous SDP) or a set of media formats that partially overlap with the previous SDP body. The offer can be rejected or accepted by the answerer. When accepted, the media format used is determined by SDP in the answer.

Overview of H.323

H.323 is a communication protocol from the ITU-T. It is a VoIP call control protocol that allows for the establishment, maintenance, and teardown of multimedia sessions across H.323 endpoints. H.323 is a suite of specifications that controls the transmission of voice, video, and data over IP networks. The following are some of the H.323 specifications relevant to the subject matter laid out in this book:

- **H.225: H.225** handles call setup and teardown between H.323 endpoints and is also responsible for peering with H.323 gatekeepers via the Registration Admission Status (RAS) protocol.
- **H.245: H.245** acts as a peer protocol to H.225 and is used to negotiate the characteristics of the media session, such as media format, the method of DTMF relay, the media type (audio, video, fax, and so on), and the IP address/port pair for media.
- **H.450:** H.450 controls supplementary services between H.323 entities. These supplementary services include call hold, call transfer, call park, and call pickup.

H.323 Components

The H.323 protocols outlined in the previous section are used in the communications between H.323 components or devices. The following are the most common H.323 devices:

- **H.323 gateways:** H.323 gateways are endpoints that are capable of interworking between a packet network and a traditional Plain Old Telephone Service (POTS) network (analog or digital). Since these H.323 endpoints can implement their own call routing logic, they are considered to be "intelligent" and, as such, operate in a peer-to-peer mode. H.323 gateways are capable of registering to a gatekeeper and interworking calls with a gatekeeper by using the RAS protocol.
- **H.323 gatekeepers:** H.323 gatekeepers function as devices that provide lookup services. They indicate via signaling to which endpoint or endpoints a particular called number belongs. Gatekeepers also provide functionality such as Call Admission Control and security. Endpoints register to the gatekeeper by using the RAS protocol.
- **H.323 terminals:** Any H.323 device that is capable of setting up a two-way, real-time media session is an H.323 terminal. H.323 terminals include voice gateways, H.323 trunks, video conferencing stations, and IP phones. H.323 terminals use H.225 for session setup, progress, and teardown. They also use H.245 to define characteristics of the media session such as the media format, the method of DTMF, and the media type.
- **Multipoint control units:** These H.323 devices handle multiparty conferences, and each device is composed of a multipoint controller (MC) and multipoint processor (MP). The MC is responsible for H.245 exchanges, and the MP is responsible for the switching and manipulation of media.

H.323 Call Flow

An H.323 call basically involves the following:

- A TCP socket must be established on port 1720 to initiate H.225 signaling with another H.323 peer. This assumes that there is no gatekeeper in the call flow. As defined in the previous section, gatekeepers assist in endpoint discovery and call admission.
- For an H.323 call, the H.225 exchange is responsible for call setup and termination, whereas the H.245 exchange is responsible for establishment of the media channels and their properties. In most cases, the establishment of two independent TCP connections is required: one for the H.225 exchange and the other for the H.245 exchange. To effectively bind the two, the TCP port number on which the answering terminal intends to establish an H.245 exchange is advertised in one of the H.225 messages. The port number can be advertised before the H.225 connect message is sent (for example, in an H.225 progress message) or when the H.225 connect message is sent.
- H.225 and H.245 exchanges can proceed on the same TCP connection, using a process called H.245 tunneling.
- Every H.245 message is unidirectional in the sense that it is used to specify the negotiation from the perspective of the sender of that H.245 message. For the successful establishment of a two-way real-time session, both H.323 terminals must exchange H.245 messages.

Figure 2-7 depicts a basic H.323 slow start call between two H.323 terminals. The calling terminal first initiates a TCP connection to the called terminal, using destination port 1720. Once this connection is established, H.225 messages are exchanged between the two terminals to set up the call. In order to negotiate parameters that define call characteristics such as the media types (for example, audio, video, fax), media formats, and DTMF types, an H.245 exchange has to ensue between the terminals.

In most cases, a separate TCP connection is established between the endpoints to negotiate an H.245 exchange; however, in some cases, as an optimization, H.245 messages are tunneled using the same TCP socket as H.225, using a procedure known as H.245 tunneling. When utilizing a separate TCP connection for H.245, the called terminal advertises the TCP port number over which it intends to establish an H.245 exchange. The ports used for the establishment of H.245 are ephemeral and are not dictated by the H.323 specification.

The H.245 exchange results in the establishment of the media channels required to transmit and receive real-time information. You should be aware that while Figure 2-7 highlights a slow start call, a variant to the slow start procedure, known as FastConnect, also exists and is depicted in Figure 2-8. As the name suggests, FastConnect is a quicker and more efficient mechanism to establish an H.323 call. In fact, FastConnect can establish an H.323 call with as few as two messages. This is possible because with FastConnect there is no need to open an H.245 socket, as long as all needed media can be negotiated via FastConnect.

Key Topic

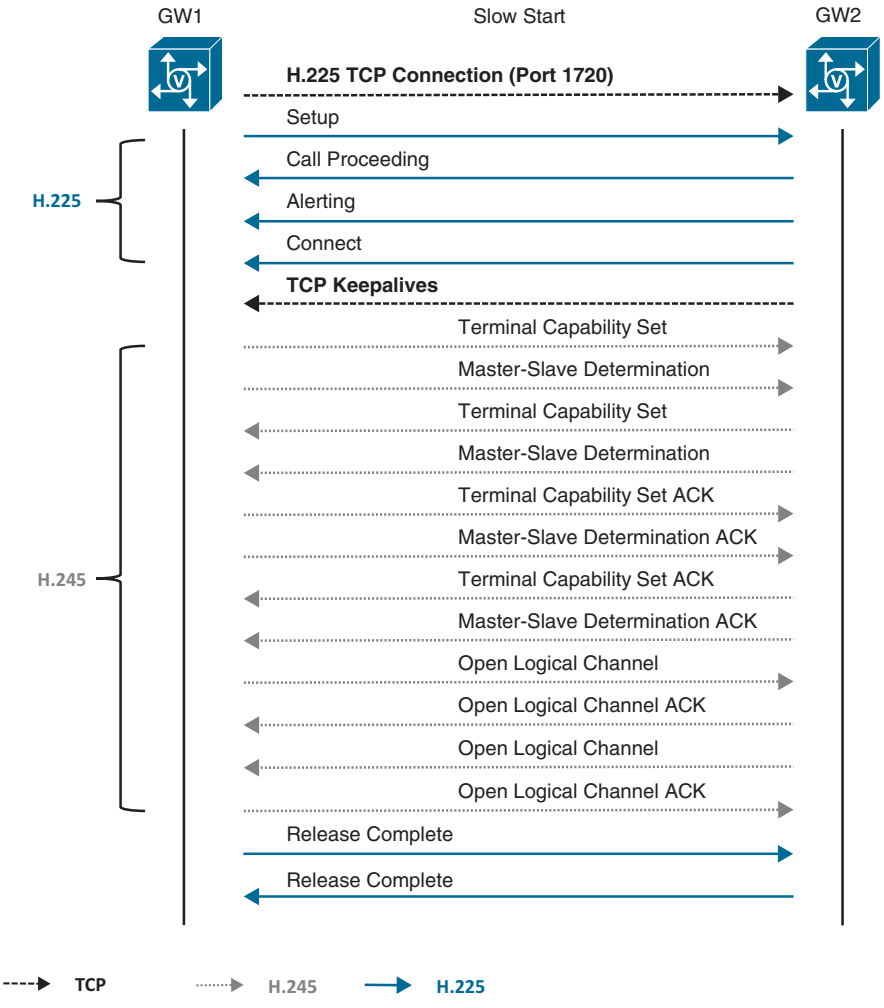


Figure 2-7 *Basic H.323 Slow Start Call*

NOTE FastConnect is often improperly referred to as "fast start" or "fastStart," after the name of the associated field/element in H.225 messages that is used to negotiate and establish FastConnect.

Figure 2-8 shows how an H.323 FastConnect call is set up. When transmitting a Call Setup message, the endpoint populates a field, known as the fastStart element, with H.245 messages. The called endpoint can accept FastConnect by selecting any fastStart element in the Call Setup message, populate the necessary data fields (as specified in H.323), and return a fastStart element in any H.225 message (for example, Call Proceeding, Alerting, Connect) to the caller. The called endpoint can also reject FastConnect and fall back to the traditional slow start procedures shown in Figure 2-7 by either explicitly indicating so (using a flag), initiating any H.245 communications, or providing an H.245 address for the purposes of initiating H.245 communications.

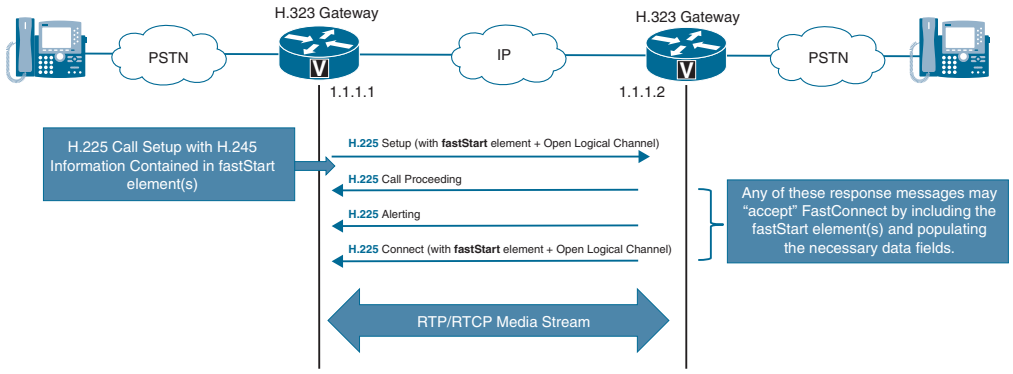


Figure 2-8 *H.323 FastConnect Call Setup*

References

Inamdar, Kaustubh, Steve Holl, Gonzalo Salgueiro, Kyzer Davis, and Chidambaram Arunachalam. *Understanding Session Border Controllers: Comprehensive Guide to Designing, Deploying, Troubleshooting, and Maintaining Cisco Unified Border Element (CUBE) Solutions*. Hoboken: Cisco Press, 2018.

RFC 3261, "SIP: Session Initiation Protocol," https://tools.ietf.org/html/rfc3261

RFC 3264, "An Offer/Answer Model with the Session Description Protocol (SDP)," https://tools.ietf.org/html/rfc3264

RFC 4566, "SDP: Session Description Protocol," https://tools.ietf.org/html/rfc4566

RFC 5853, "Requirements from Session Initiation Protocol (SIP) Session Border Controller (SBC) Deployments," https://tools.ietf.org/html/rfc5853

Exam Preparation Tasks

As mentioned in the section "How to Use This Book" in the Introduction, you have a couple of choices for exam preparation: Chapter 11, "Final Preparation," and the exam simulation questions in the Pearson Test Prep software online.

Review All Key Topics

Review the most important topics in the chapter, noted with the key topics icon in the outer margin of the page. Table 2-9 lists these key topics and the page number on which each is found.

Table 2-9 Key Topics for Chapter 2

Key Topic Element	Description	Page
Figure 2-1	Sample SIP Dialog with Two Transactions	32
List	List of SIP components	34
Table 2-2	SIP Requests	35
Table 2-3	SIP Response Classes	36

Key Topic Element	Description	Page
Figure 2-3	Analyzing a Basic SIP Call	46
Table 2-5	Fields and Attributes in SDP Bodies	50
Figure 2-4	SIP Delayed Offer Versus SIP Early Offer	53
Table 2-7	SDP Direction Attributes Description	56
Table 2-8	Different Ways of Setting the Direction Attribute in an Answer	58
Figure 2-6	SIP re-INVITE and UPDATE Session Modifications	61
List	H.323 components	67
Figure 2-7	Basic H.323 Slow Start Call	69
Figure 2-8	H.323 FastConnect Call Setup	70

Complete Tables and Lists from Memory

Print a copy of Appendix C, "Memory Tables" (found on the companion website), or at least the section for this chapter, and complete the tables and lists from memory. Appendix D, "Memory Tables Answer Key" (also on the companion website), includes completed tables and lists to check your work.

Define Key Terms

Define the following key terms from this chapter and check your answers in the glossary:

Session Initiation Protocol (SIP), Session Description Protocol (SDP), user agent client (UAC), user agent server (UAS), back-to-back user agent (B2BUA), H.323, H.225, H.245

Index

Numerics

A

B

C

D

E

F

G

H

I

J-K

L

M

N

O

P

Q-R

S

T

U

V

W

X-Y-Z