Compliments of Sonus

2nd Sonus Special Edition

WebRTC

FOR DUMMIES®

A Wiley Brand

Learn to:

- Master the basics of WebRTC
- Integrate WebRTC with your current infrastructure
- Leverage WebRTC in real-world applications and services
- Identify the must-haves when deploying WebRTC

Justin Hart

WebRTC

FOR DUMMIES®
A Wiley Brand

2nd Sonus Special Edition

By Justin Hart

FOR DUMMIES®
A Wiley Brand

WebRTC For Dummies®, 2nd Sonus Special Edition

Published by
John Wiley & Sons, Inc.
111 River St.
Hoboken, NJ 07030-5774
www.wiley.com

ISBN: 978-1-119-09878-2 (pbk); ISBN: 978-1-119-09885-0 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Publisher's Acknowledgments

Some of the people who helped bring this book to market include the following:

Project Editor: Carrie A. Johnson
Editorial Manager: Rev Mengle
Acquisitions Editor: Katie Mohr
Business Development Representative: Sue Blessing
Production Editor: Vinitha Vikraman

Table of Contents

Foreword

Welcome to the 2nd Edition of *WebRTC For Dummies*. As a long-time observer and analyst of the technology, I believe it has the potential to become truly transformative. There could be as many as 6 billion devices supporting WebRTC by 2019, and as many as 2 billion people using it — over half the world's Internet population. Books such as this, which introduce key ideas to new audiences, are important in helping it follow that trajectory.

WebRTC is a major step toward what is called "contextual communications," moving voice and video from generic, standalone "calls" to events that are embedded into websites and mobile apps. While it is good to have the ubiquitous phone or a videoconference system as a "jack-of-all-trades," there are many instances where we could benefit from combining a conversation with the web.

For example, imagine directly conducting a recruitment video-interview "inside" an HR application, with recordings made securely-available for everyone in the hiring process to view later. Perhaps a remote telemedicine call could allow a doctor to both speak to the patient and simultaneously browse a website showing health and nutrition advice. An existing contact center might wish to add a special "video concierge" service for its premier customers, starting with just a few agents but perhaps expanding over time.

It is the diversity of use-cases that makes WebRTC so powerful — it can enhance or extend current systems owned by businesses or telcos, or allow entirely new services and platforms to be created. It has uses in B2B, B2C, C2C, and maybe even M2M worlds. It might appear in an oilfield video-maintenance application or enhance a new form of mobile social-media app for teenagers.

WebRTC is also easily componentized (or "delaminated"), allowing developers who are creating a communications-enabled app to choose what to build in-house, where to

integrate with existing assets, and what to obtain from various cloud providers. Signaling, media-handling, client frameworks, analytics, security, interoperability, dealing with firewalls can all be dealt with individually or in convenient packages, either in hardware, software, or virtualized in the cloud.

However, it is not without its practical challenges, too. The standards are still evolving, and while they are maturing fast, continual enhancements can be expected as well. Many customers and developers — especially enterprises and telcos — will want to re-use their existing systems and capabilities: SIP-based UC, familiar numbering, core networks, and so forth. Some of these need to be interworked with WebRTC quite carefully, bridging between the two worlds securely and simply. They may also need to work out how to support users with older browsers — or those on mobile devices for whom native apps are often better-optimised for communications than the web.

The WebRTC industry has to work out how to bring two worlds together: communications experts adding "the web" and web experts adding "communications." Often, their language and expectations are different — and certainly their skillsets. There are a lot of great sources of WebRTC technical information around, but they are often intimidating for people from other backgrounds. Against that background, I applaud the availability of books such as this, which help widen the community of people who can understand both the technical and business implications of this disruptive technology.

Dean Bubley

Director of Disruptive Analysis

@disruptivedean

London, March 2015

Introduction

Web Real-Time Communications (WebRTC) is a new kind of communications that's already generating a lot of industry talk — and for good reason. WebRTC has the potential to transform the way you communicate by enabling voice, video, collaborative sessions, and other real-time communications from any device that supports a standard web browser. With no plug-ins needed and no download of client software necessary, WebRTC promises to open up real-time communications to a whole new world of possibilities.

Note: Despite its enormous potential, WebRTC is still in its relative infancy. Industry players from standards organizations to commercial web browser companies are still deciding how best to implement WebRTC for the whole world, even as enterprises and service providers decide when and where WebRTC will have the most value for their customers. Most agree, however, that WebRTC may be a game-changer for its capability to deliver high-quality, low-cost, ubiquitous communications.

About This Book

WebRTC For Dummies, 2nd Sonus Special Edition, is written to help you understand what WebRTC is, how it works (and what you need to make it work), and why it's important in an enterprise's or service provider's communications ecosystem. This book is written for the business layperson; if you're looking for source code or architectural network diagrams, look elsewhere (and trust me — plenty of highly technical books on the subject are available).

Although I'm going to talk a lot about voice and video communications in this book, it's important to remember that WebRTC isn't limited to voice or video communications alone. WebRTC also supports real-time applications such as collaborative applications (such as screen sharing and online whiteboarding) and even online gaming.

Icons Used in This Book

This book calls out important bits of information with icons on the left margins of the page. The icons that you see in this book include the following:

The Tip icon points out a bit of information that aids in your understanding of a topic or provides a little bit of extra information that perhaps isn't 100 percent necessary, but which may broaden your understanding of what you've just read.

The Remember icon alerts you to useful information that will come in handy later.

Watch out! This icon tells you to steer clear of things that could cost you big bucks, delay your project, or represent bad practices.

Chapter 1

Getting Acquainted with WebRTC

In This Chapter

- Looking at the support system for WebRTC
- Understanding how WebRTC works
- Knowing why WebRTC is important
- Identifying the different parts of WebRTC
- Realizing the real power of WebRTC with the Data Channel API

It wasn't long ago that the idea of communications evoked the single image of two people talking on a telephone connected to the wall. Then came wireless phones, mobile phones, softphones, smartphones, tablets, smartwatches, and so on. Today the idea of dialing someone on the phone seems as antiquated as cranking up an automobile for a Sunday drive.

The rapid evolution of communications has been driven by two key benefits: mobility and simplicity. Consider, for example, the way you connect to the world around you. In the past, you had to dial a person's phone number, often with the aid of a phone book that was heavy enough to crack a walnut. Then came contact lists, click-to-dial, instant messaging (IM), and even instant video chats from your phone. With the introduction of Unified Communications (UC), users can now manage all these different communications methods from a single screen on any device — the very model of mobility and simplicity.

Still, these communications require specialized software to make them work. For example, IM chats require an IM client on the device, video chats require a video app, and voice calls require a voice service provider and (of course) a phone

number. But all that is about to change thanks to a new technology known as Web Real-Time Communications (WebRTC).

WebRTC is a new communications model that delivers voice, video, and other real-time communications (such as screen sharing, whiteboarding, online gaming) with no additional client-end software beyond a standard web browser — something that nearly all devices already have. In other words, it opens the door for live, end-to-end communications from nearly any endpoint: phones, web pages, kiosks, automobiles, and even household appliances. If it can support a web browser, WebRTC can turn it into a real-time communications device — no service provider needed (beyond an Internet Service Provider — ISP) and no number (beyond an IP address) required.

The applications for WebRTC are almost limitless. For example, WebRTC would allow customers visiting your website to open a real-time video chat with a customer service representative or open a video chat with a friend to share the page ("Honey, what do you think about this house?"). It could also be used to embed real-time customer service in bank ATMs (through a "speak to a live agent now" button, for example) or to enable person-to-person chat sessions without sharing phone numbers or other personal information (for example, in the case of an online counseling hotline). The simplicity and mobility of WebRTC make it an ideal communications platform for a world that loves to communicate but can't always agree on which application/device to use for that communication.

Who Supports WebRTC Today?

Google was one of the primary founders of WebRTC, beginning with its open-source implementation of WebRTC in 2011. Since then, Internet standards groups such as the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C) have taken charge of the effort to advance WebRTC. A host of other software and hardware companies, notably Cisco and Microsoft, and other browser vendors, including Mozilla, the maker of Firefox, have also joined the effort. (Although Apple hasn't announced WebRTC support for iOS at the time of this writing, several third-party applications provide WebRTC support for the Safari browser.)

See Figure 1-1 for a view of how the web has evolved and how WebRTC plays a part in this evolution.

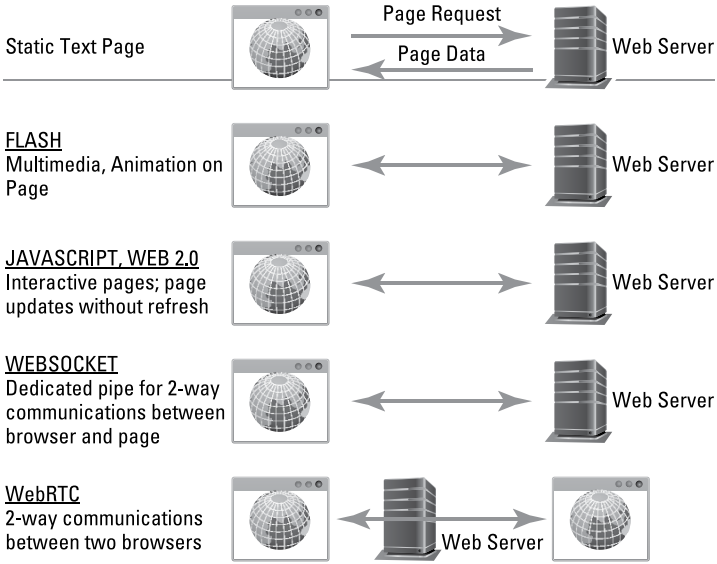

Figure 1-1: Evolution of the web.

How Does WebRTC Work?

WebRTC isn't a solution like UC; instead, it's a communications infrastructure. To use a familiar analogy, you can think of WebRTC as a road and UC as a car on that road. The fact that WebRTC is an open-source technology (it's available to everyone for free) and uses the common JavaScript language means that millions of web developers are able to build WebRTC applications with relative ease. In fact, you could say WebRTC is as easy as 1-2-3-4:

1. **A web developer uses JavaScript (and minimal HTML5) code to embed WebRTC functionality in a website or application.**

   This functionality could take the form of a "chat now" button or "share live with a friend" link, for example.

2. **A user simply clicks the button or link to launch a WebRTC session.**

3. **WebRTC opens a session request between two endpoints without sharing any personally identifiable information — no name or number is required,**

although WebRTC can display that information if desired.

4. **If the session is accepted by the receiving endpoint, WebRTC opens a live, real-time communications session that's displayed/conveyed via the web browser.**

   That's it. No additional client-side software or plug-ins need to be installed.

WebRTC makes it possible for two users to communicate with each other even if they're using different devices, different operating systems (Android, Windows, Linux, and so on), or even different browsers.

Why is WebRTC Important?

The simplicity and potential ubiquity of WebRTC make it an ideal communications platform for enterprise applications, particularly in the area of customer service. Many enterprises today have already integrated IM chat and click-to-call features into their online customer experience. With WebRTC, enterprises could extend those capabilities to include video and collaborative sessions. For example, a retail business may use WebRTC to create an interactive session between a customer service agent and a customer as they browse the retailer's website. Or a bank could use WebRTC to provide a voice/video chat link to a high-value customer's private banker right on the customer's online account page — and connect the customer to the private banker immediately with one click.

WebRTC also has great potential for "emergency" situations, from home repair to healthcare. The capability to instantly open a video chat within a web browser or customer service app means that customers can quickly elevate the discussion to share vital visual information with experts on the other end of the call.

You may be wondering how WebRTC fits in with other enterprise communications. As enterprises upgrade or replace their traditional, siloed communications tools for integrated or unified systems, they're looking to deliver a simplified experience around mobile devices such as the smartphone. The beauty of WebRTC is that it takes away the requirement for specialized client hardware (for example, devices) and

software. It even removes the need for browser plug-ins. Now the browser, which all Internet-connected devices already have, can be the interface for UC.

Within an enterprise, this is handy, but it becomes a really big deal when you think of how an enterprise communicates outside of its walls. For example, what if employees of an enterprise need to have a videoconference with key suppliers or with partners who are consulting with the company on a big project? With WebRTC, they can simply direct them to a web page, and voila! They're part of the videoconference.

The examples given are already possible today with UC, but they require either some client software or special browser plug-ins to make them work. With WebRTC, none of that's needed. So long as users have the mobile app or access to a compliant web browser, they're ready to connect right now — and nearly all popular web browsers either support WebRTC today or plan to do so in the near future. WebRTC makes accessing UC services easy, painless, and foolproof.

Understanding the Elements that Make Up WebRTC

Although WebRTC is easy, painless, and foolproof from the end-user's perspective, there's a fair amount of technology working behind the scenes to bring it all together. In this section, I introduce some of the key technology components that support WebRTC communications.

Browsers

One of the primary elements of WebRTC, at least from the user's perspective, is the web browser. Organizations that create web browser software need to incorporate certain elements such as the WebRTC Application Programming Interface (API) into their browser code to allow the browser to control the elements of the computer's hardware and software required for WebRTC communications.

Many of the most popular web browsers already support WebRTC, including the latest versions of Google Chrome,

Mozilla Firefox, and Opera. Although Microsoft Internet Explorer (IE) doesn't support WebRTC, Microsoft is definitely engaged in the WebRTC space but around a slightly different implementation of WebRTC known as Object Real-Time Communications (ORTC). It remains to be seen how WebRTC and ORTC will evolve together, but the two technologies are on a parallel path and will likely converge in the near future.

WebRTC is based on JavaScript, which is the programming language used to create many client-side scripts for interactive web apps, and the latest version of the HTML markup language, HTML5. JavaScript and HTML5 enable web browsers and mobile apps to present things such as audio and video interactively, much like the Adobe Flash plug-in for web browsers. WebRTC simply extends the capabilities of HTML5 and JavaScript into the field of real-time communications.

Web browser APIs

Web browsers support WebRTC through JavaScript-based APIs that enable hardware control and communication through the browser. The most important APIs behind WebRTC-compliant browsers include the following:

- **getUserMedia:** This API allows the web browser to control video and audio hardware such as a webcam or microphone.
- **RTCPeerConnection:** This API helps to set up the WebRTC communications, much as touch-tone dialing does for traditional analog phone calls. PeerConnection goes beyond making the connection; however, it's also involved in elements of the call such as choosing codecs and applying encryption.
- **RTCDataChannel:** DataChannel is the data portion of the WebRTC communication. DataChannel provides a peer-to-peer, decentralized data connection between the browsers through which the data can flow directly between endpoints in real time. (I talk more about the DataChannel API at the end of this chapter in the section "Understanding the Data Channel.")

Web servers

Another important element of the WebRTC environment is the web server that hosts the JavaScript and HTML code. This server will (ahem) serve up the clickable URLs in a web page that a WebRTC user interacts with when initiating a real-time communications session. This web server could be integrated into a gateway or located on an existing web server within an enterprise's or service provider's network.

Codecs

A *codec* — a combination of "coder-decoder" — is a computer program that's used to encode a signal for storage and transmission and then decode it at the far end for display or playback. WebRTC codecs encode/decode audio and video signals so users can listen to and view audio/video calls from their web browser or app.

Nearly all the digital audio or video content you interact with via a call on your mobile phone, a YouTube video you're watching on your iPad, or the digital HDTV signal that's displayed on your big-screen TV uses a codec for the encoding and decoding process. In WebRTC, several "official" codecs for audio and video are involved: for audio, Opus and G.711; for video, VP8 and H.264.

Each codec has its advantages. G.711, for example, is easy to use, while Opus is a royalty-free codec. And there is still lively discussion in the broader WebRTC community as to which codecs should be included in WebRTC implementations, particularly around video codecs. For the time being, network elements such as a media resource function (MRF) or media server will likely be required to transcode (that is, translate codecs) between the different choices.

Web applications

WebRTC includes a Web API designed to allow developers to create their own WebRTC-enabled applications. An organization implementing WebRTC can create its own applications or purchase them from software vendors. For example, there isn't just one WebRTC videoconferencing application or one

WebRTC collaboration application. Instead, third-party organizations can create their own WebRTC applications by using the Web API and WebRTC specifications.

Gateways

To open a voice or video call between a user on its website and an internal employee, an enterprise can either set up a WebRTC-to-WebRTC (or peer-to-peer) communication or integrate the WebRTC communication into its existing back-end communications system. Most corporate communications systems use Session Initiation Protocol (SIP) to connect to their audio and video systems. Therefore, to connect a WebRTC communication to a corporate communications system — be it a call center application or a UC solution — the WebRTC communication has to be translated into SIP.

The device that makes the connection between WebRTC and an organization's communications network is called a *gateway*. The gateway bridges the divide between WebRTC and the enterprise communications network (or a service provider network if the communications services reside there). The gateway does this by performing the interworking between SIP and WebRTC. (I discuss interworking in more detail in Chapter 2.)

Session Border Controller

Another important component of the enterprise and service provider network is the *Session Border Controller* (SBC). The SBC is a device deployed on the border of enterprise and service provider Voice over IP (VoIP) networks where the internal part of the network connects to external networks. The SBC has several key functions in the network:

- **It protects the internal network.** SBCs are designed to protect enterprise and service provider networks from a variety of malicious threats such as Denial of Service (DoS) attacks, toll services theft, and spoofing. In a WebRTC environment, an SBC secures an organization's internal VoIP network from web-based attacks coming from WebRTC users on the network.

- **It protects communications beyond the network.** SIP-based communications are often encrypted by using a protocol known as *Secure Real-time Transport Protocol* (SRTP). Legacy SIP applications use a version of SRTP that's currently incompatible with WebRTC, so an SBC is required to encrypt/decrypt SRTP-enabled communications between legacy SIP and WebRTC endpoints.
- **It controls calls coming in to the network.** The SBC is the "gate guard" for an enterprise or service provider network, determining which calls and communications sessions should be allowed onto the network. This screening can include the use of blacklists (always blocked calls) and whitelists (always admitted calls). When traffic volumes are particularly high, the SBC's call admission control feature helps to make the network more secure and reliable. An enterprise can therefore use an SBC to help ensure that only legitimate calls from WebRTC users are entering its VoIP network.
- **It translates between different codecs.** The SBC can transcode between codecs and adjust the bit rate (known as *transrating*) between both ends of a WebRTC call. WebRTC may use different codecs than an enterprise or service provider VoIP network, so the SBC allows the network to do on-the-fly conversions between codecs so both sides can "talk" to each other.
- **It provides interworking with enterprise networks.** SBCs are really good at getting calls through to the actual endpoint within an enterprise network. This process isn't a simple matter when you consider that many enterprises use Network Address Translation (NAT), which intentionally "hides" private IP addresses within the enterprise network behind a single "public" IP address facing the outside world. SBCs provide *NAT traversal* so calls can easily reach the right device — be it a computer or an IP phone — in the enterprise network. The SBC helps connect calls from WebRTC users within an enterprise or service provider VoIP network by using a variety of NAT traversal techniques that include Interactive Connectivity Establishment (ICE) and Session Traversal Utilities for NAT (STUN).

SBCs can also hide the topology of an enterprise network from the outside world. The result is a network that's easily accessible to clients for the purpose of making and receiving calls, yet where the "innards" of the network are effectively invisible, making network devices and endpoints less vulnerable to attack.

Understanding the Data Channel

A particularly neat feature supported by WebRTC is the DataChannel API. DataChannel provides a peer-to-peer connection — meaning that there's no centralized server controlling the flow of data between browsers — for data beyond just audio and video.

A big advantage of cutting out the middleman (for example, the server) and going peer to peer is that connections can be extremely fast and latency kept low (*latency* is the delay in the transmission of data between two browsers).

The DataChannel API opens up all sorts of possibilities for WebRTC in terms of services such as web collaboration and screen sharing, online gaming, and file sharing. In other words, WebRTC, through the DataChannel, supports more than just voice and video, right in the web browser. DataChannel opens up a huge range of new applications without requiring client software.

DataChannel supports two types of connections:

- **Reliable channels:** These channels provide the highest level of assurance, ensuring data is received at the far end of the connection and retransmitting it when it isn't. With a reliable channel connection, data will be received in the order it was sent and potentially delayed if a retransmission is required.
- **Unreliable channels:** These channels limit the re-transmissions and don't necessarily ensure that data is received in the order it was re-transmitted.

Chapter 2

Integrating WebRTC into Other Networks

In This Chapter

- Finding out where you can connect with WebRTC
- Looking at the reasons you need an SBC
- Utilizing a WebRTC gateway

In Chapter 1, I introduce Web Real-Time Communications (WebRTC) and cover some of its basic building blocks. These building blocks help you understand how WebRTC provides voice, video, and other communications services directly from a web browser or mobile app. In this chapter, I talk about how WebRTC integrates with the existing Voice over Internet Protocol (VoIP) or Unified Communications (UC) network that an enterprise or communications service provider may already have in place.

It is, of course, possible for WebRTC to be used for browser-to-browser (or app-to-app) communications, which bypasses the communications network of an enterprise or service provider entirely, but some organizations will still choose to integrate WebRTC into their existing communications infrastructure. Integrating WebRTC into an existing network makes it simpler for people outside the network (for example, customers) to call, conference, and video chat with people inside the network using nothing more than a standard web browser.

Integrating WebRTC with your existing communications networks is more than just a useful way to expand the reach of your network; it's also a great way to extend your VoIP network's service life and keep it from becoming obsolete.

Companies that have invested huge amounts of money in their VoIP networks can easily extend the useful life of those networks by integrating with WebRTC as it becomes more prominent. There's no need to rip and replace your old VoIP network. Instead, you can augment your legacy VoIP network to support WebRTC calls.

Learning Where WebRTC Can Fit into Your Network

Before we begin digging into the bits and pieces required to integrate WebRTC into an existing VoIP network, it's worth looking at what services this integration can provide. An enterprise or service provider can choose to integrate WebRTC services into its existing networks to do the following:

- **Add WebRTC functionality to an existing Session Initiation Protocol (SIP)-based network.** For enterprises, the most obvious benefit of this point is the capability to add WebRTC functionality to an existing call center infrastructure. Integration allows anyone with a browser or customer service app and an Internet connection to contact a company's customer service or sales department without needing to know phone numbers or having to download a specific application or plug-in.
- **Integrate with the Public Switched Telephone Network (PSTN).** Another good use case is to enable employees who aren't at a work location to use a WebRTC interface to make phone and video calls. Doing this can eliminate long-distance and mobile roaming charges or, at the very least, reduce those charges by leveraging an enterprise's existing SIP-based wide area network (WAN). Using a WebRTC interface means that remote workers don't have to install another app on their devices or worry about computer settings and compatibility — a boon for enterprises that have adopted a Bring Your Own Device (BYOD) strategy for communications.
- **Provide easy external access to a UC system.** Enterprises that have deployed a UC platform, such as Microsoft Lync or the new Skype for Business, can easily expand the reach of that platform by integrating it with WebRTC.

This allows remote users, mobile users, partners, and customers to access a rich range of UC services such as videoconferencing and real-time collaboration through a simple (and lower cost) web browser connection.

- **Provide access to telecommunications services.** Telecommunications service providers can leverage WebRTC to provide access to their portfolio of voice, video, and related services beyond their existing network-based subscribers. For instance, a service provider could offer special calling or video services such as discounted international calls from WebRTC clients to phones on the remote end of the call without requiring the customer to be a current subscriber to the telco's services. Additionally, service providers could use WebRTC to extend access to the applications and services they run in their cloud so users can take advantage of those communications features from anywhere.

Understanding the Need for an SBC

The Session Border Controller (SBC), which I introduce in Chapter 1, is a mainstay in enterprise and service provider VoIP and UC networks. At the border between internal and external networks, the SBC provides important services:

- Secures the network
  - Protects against Denial of Service (DoS) attacks
  - Hides the topology of the internal network from external view
  - Encrypts signaling and media (while also supporting Legal Intercept of calls where required)
  - Guards against spoofing and other malicious attacks
- Provides interoperability
  - Transcodes and transrates media
  - Translates between the many different varieties of SIP (also known as *SIP normalization*)

- Implements and enforces network policies (covered in more detail later in this chapter under "Managing policies")
- Provides a platform for creating new SIP-based services

Many enterprises and service providers already use an SBC to perform these functions today. The good news is that SBCs can bring a lot of these functionalities over to the WebRTC world by acting at the point where WebRTC calls interface with the VoIP or UC network. In the remainder of this section, we discuss where and how WebRTC and SBCs specifically work together in this environment.

You don't need an SBC for WebRTC-to-WebRTC sessions where no communications network is involved (for example, a gaming application between two PCs). In the enterprise and service provider world, however, where WebRTC is integrated with an existing SIP or UC network, the SBC has a vital role to play.

Interworking for interoperability

One of the biggest roles that an SBC plays in supporting WebRTC in an existing SIP or UC network is through *interworking,* which is the allowing of different and often incompatible systems, media, and protocols to communicate, connect, and exchange data.

WebRTC and SIP/UC networks both use VoIP, but they don't use VoIP in exactly the same way (case in point: check out the Chapter 1 discussion of incompatibilities found in SRTP encryption). Although most UC networks utilize SIP, WebRTC uses, well, WebRTC. And most SIP/UC systems use additional or different audio and video codecs (I cover these in Chapter 1) than those employed by WebRTC.

For a WebRTC client (for example, a browser) to communicate with a SIP-based client such as a smartphone or softphone, something needs to make them speak the same language. The SBC can do that by providing the following:

- **Transcoding:** Real-time conversion between WebRTC and SIP audio and video codecs; for example, translating the WebRTC Opus codec to a standard SIP codec like G.711.

- **Transrating:** The SBC can also *transrate* (or modify the bit rate and required bandwidth) of multimedia sessions, as dictated by network demands. For example, if a WebRTC call is placed from a computer on a gigabit Ethernet connection and ultimately ends up on a mobile device with limited 3G bandwidth, the SBC can automatically reduce the bandwidth of the call to fit the available "pipeline" on both ends of the call.

Securing the SIP network

WebRTC voice and video calls are placed and received on PCs and other devices, such as tablets and smartphones, which are connected to the Internet. Chances are you're already well aware of the security risks inherent in Internet-based communications — after all, there's a reason you have to constantly update your anti-virus and anti-malware programs.

As you may suspect, a possibility that a compromised endpoint will attempt to connect to an enterprise or service provider's SIP network via WebRTC always exists. Scenarios like this keep CIOs up at night, especially as they consider opening up their business-critical call centers to WebRTC calls.

An SBC can greatly reduce this risk by

- Protecting against DoS attacks that could tie up call center resources with a flood of simultaneous calls
- Providing call admission control via whitelists, blacklists, and other policies in order to regulate call flow and block undesirable calls
- Blocking rogue WebRTC requests that could harbor a malicious network attack
- Hiding the topology of the internal SIP network — that is, rendering internal devices and network elements invisible — from WebRTC clients and the wider web

Navigating the firewall

Firewalls provide unauthorized access to data within the enterprise or service provider network, much as an SBC provides unauthorized access to communications media.

In order to facilitate WebRTC communications between endpoints on different sides of a firewall (for example, inside and outside the firewall), signaling information such as port and IP address needs to pass through the firewall to establish the communications connection. This requires support for specialized technology such as Interactive Connectivity Establishment (ICE) on the part of the SBC or gateway device.

ICE is a standard, secure procedure that defines the method for determining the public IP address and port information for external and internal users. To accomplish this, ICE uses two protocols:

- **STUN:** Session Transversal Utilities for Network Address Translation (NAT)
- **TURN:** Traversal Using Relay NAT

If neither WebRTC user is behind a firewall or NAT, ICE isn't required. However, wherever a firewall exists, ICE/STUN/TURN is required to enable communications.

Many freeware-based ICE/STUN/TURN applications are available on the market today. These capabilities are also frequently built into gateways and SBCs. Users don't have to worry about this, though; the process is handled behind the scenes by the communications and network teams at the company or service provider when the WebRTC-based services are set up.

Managing policies

Enterprise and service provider networks are built on top of policies. What we mean by that is that these networks aren't "free for alls" where anything goes and all clients can use every service whenever they want. Like any network, VoIP and UC networks aren't limitless resources; policies exist to determine who can access what resources and when.

One of the primary guardians of network policy is the SBC. The SBC is the "traffic cop" of the communications network, providing call access control and determining which calls are allowed on and off the network in the first place.

There's more to policy than call admission control. Policy also includes decisions on things such as

- Bandwidth utilization and rate limiting
- Least cost routing for toll calls
- Application availability
- Media paths and routing

Because of its low-latency and peer-to-peer nature, WebRTC can enable rich, high-bandwidth multimedia communications. Policy is what determines how that demanding media will be handled when it crosses the border into an enterprise or service provider's network. Policy also helps ensure that service level agreements (SLAs), which determine the minimum quality levels that should be maintained in a VoIP/UC network, are met.

Finally, policy helps determine what happens when the SIP network gets congested — for example, when there's a great surge in calls (which Marketing will probably take the credit for). Policy, as implemented through the SBC, can determine if certain services need to be throttled back or turned off or if call overages need to be directed elsewhere.

Figuring Out the Role of WebRTC Gateways

There's really only one role for a WebRTC gateway, but it's a big one: interworking. If the SBC is the traffic cop securing the border between your VoIP/UC network and WebRTC clients, then the WebRTC gateway could be seen as the bridge between the two network segments that allows communications to flow from one side to the other.

Figure 2-1 shows the configuration of an enterprise network with the SBC and the WebRTC gateway intermediating between WebRTC and an enterprise's internal network.

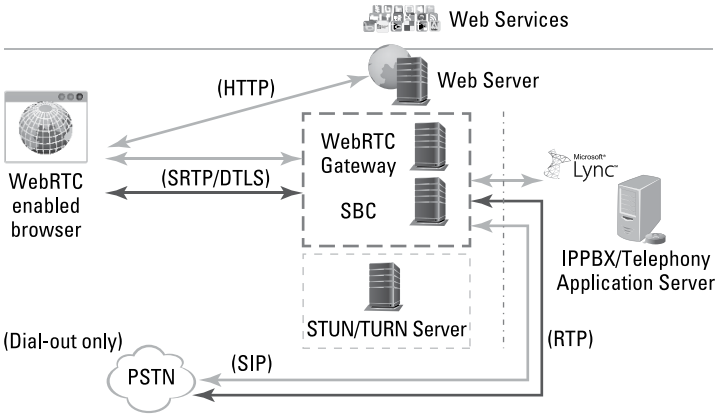

Figure 2-1: The gateway and SBC in an enterprise network.

A WebRTC gateway converts WebRTC and SIP signals so calls can be connected, carried, and ended. This important role also includes the basic routing of the signaling and media parts of the calls across different VoIP network segments.

Chapter 3

What Can WebRTC Do for Me?

In This Chapter

- Connecting contact centers to customers
- Making videoconferencing easy
- Extending IMS networks for service providers
- Expanding the reach of OTT services
- Getting into gaming

This chapter is a very interesting one — or at least I think so — because I talk about what you can *actually* do with Web Real-Time Communications (WebRTC). The use cases describe the power of WebRTC — I think you'll enjoy reading them as much as I did writing them. This chapter shows you that WebRTC has the potential to be a real game-changer.

Calling All Enterprise Connect Centers!

As far as enterprise applications for WebRTC go, this one is a no-brainer. Most mid-to large-sized enterprises (and government agencies) have already adopted Session Initiation Protocol (SIP)-based Voice over Internet Protocol (VoIP) systems for their contact centers. Whether the application is customer service, inbound/outbound sales, or some other customer-facing activity, chances are good that a customer call will soon find its way to a SIP-connected call center.

The popularity of SIP is simple: It's much cheaper to operate a call center using SIP than in a traditional Time Division Multiplexing (TDM)-based environment, and it's much easier to manage a geographically dispersed call center with VoIP.

Recently, enterprises have begun to integrate click-to-call services on their websites that expand the reach of call centers to online customers. These services, however, haven't always been so simple for the end-user because they required steps such as entering a call back number or downloading a browser plug-in. WebRTC takes that complexity away. Now, any end-user with a compliant browser or a WebRTC-enabled app can simply click on a link and instantly connect via audio or video to the call center.

Yes, I said connect via *video*. WebRTC makes it easy to add a video chat component to any contact center, which elevates the customer experience and can have a significant impact on customer satisfaction and revenue. It can also have a significant impact on your dress code (no more Heavy Metal T-Shirt Tuesdays), but that's a different story.

Facing the Future with Videoconferencing

Thanks to videoconferencing, a face-to-face meeting no longer means corralling everyone into the same room. Many enterprises have adopted videoconferencing as a way to deal with their increasingly long-distance workforce. Whether it's supporting teleworkers, remote offices, road-warrior sales teams, or that newly acquired division in Tokyo, videoconferencing allows for richer and more immersive collaboration across the miles. We've even seen a company have some fun with this by taping a videoconferencing-equipped iPad to the head of a dummy seated at its conference table!

But there's still a catch with videoconferencing: *compatibility*. Getting every employee to use the same videoconferencing application is relatively simple thanks to Unified Communications (UC) platforms like Microsoft Lync (now Microsoft Skype for Business). Getting consultants, partners, and suppliers to join in the same videoconference can be

harder because every business has its own IT infrastructure and may use a completely different communications platform.

WebRTC takes that problem right out of the picture. Businesses can use whatever platform they want (including WebRTC videoconferencing apps); all you need to join a videoconference in WebRTC is a mouse or touch-sensitive screen. And because WebRTC doesn't require special hardware or software, it significantly reduces the cost of videoconferencing in your business.

Bringing a Carrier's IMS Network to the Web

Telecommunications service providers (or *telcos*) have spent the better part of the past decade developing an architectural framework for next-generation networks known as the *IP Multimedia Subsystem* (IMS). IMS is designed to facilitate the provisioning and delivery of IP-based communications services (such as VoIP) and future services (such as multimedia mash-ups of audio and instant messaging) using standards-based protocols (for example, SIP and Diameter).

IMS is the basis for many of the 4G Long Term Evolution (LTE) services being offered to telcos' mobile customers today, as well as the IP-based services offered to fixed-line, broadband customers. IP Multimedia Subsystem (IMS) and UC services both use IP standards such as SIP, so they're compatible and similar in a lot of ways. 3G and 4G LTE phones can use SIP-based IP calling, whereas earlier-generation mobile phones were limited to Global System for Mobile (GSM) communications or Code Division Multiple Access (CDMA) calling.

By integrating WebRTC with IMS, many of the services offered to telco endpoints, such as IP phones, smartphones, and tablets, can be extended to any web browser anywhere in the world. With a WebRTC dialing browser Application Programming Interface (API), for example, you could place calls to mobile phones directly from a browser without the need for client-side software or a browser plug-in. (Currently, the industry group that manages the IMS standard, 3GPP, is still hashing out the details of an architectural framework for WebRTC integration with IMS, so stay tuned.)

Extending Over-the-Top Services

One of the more interesting telecommunications trends in recent years is the proliferation of *over-the-top* (OTT) services. The name originates from the fact that these services are delivered on top of an existing network connection (usually broadband) instead of through the network service providers themselves. For example, Skype is an example of an OTT communications application, while Netflix is an example of an OTT video streaming application.

Hundreds of OTT services are available today, and they serve both the consumer and business markets. Many OTT services are voice-, audio-, or video-based applications. One thing they all tend to have in common — in addition, of course, to being delivered over the top of an existing Internet connection — is that they require some sort of software download. Skype requires an app download to a PC or smartphone/tablet. Netflix requires an app on mobile devices or an installation of Microsoft Silverlight to work on PC web browsers. And once you require downloads and installations, you also require users to install update and keep doing so over time. What a drag.

With WebRTC, there's no need to install a dedicated application just for communication. In fact, you only have to install the latest web browser (which in many cases will automatically update itself). By leveraging WebRTC, OTT service providers can improve their offerings by making them accessible to more people.

Technology is a two-way street, however, as WebRTC may one day help network operators, such as telcos and cable providers, effectively compete with OTT providers. In a WebRTC environment, for example, a telco could quickly and easily build OTT-like applications, deliver them via existing web browsers, and then use its sophisticated IMS networks to deliver higher levels of service assurance, interoperability, and performance than a "pure play" OTT provider could offer.

Games, Games, Games!

Whoever said work and play don't mix should consider this: In 2013, according to the Entertainment Software Association, the gaming industry generated $21 billion in sales in the U.S. alone!

Online, multiplayer gaming represents a big portion of that market, especially among hard-core gamers — whether they're chasing each other around a track on *Mario Kart* or chasing aliens on a distant planet in *Halo.* Typical multiplayer games rely on a central server, and performance is dictated by not only Internet connectivity but also by server capacity and utilization. The gaming community has even invented an acronym for people who have superior connectivity: LPB. (The first two letters stand for Low Ping, which refers to the low latency in their Internet connections, while the B stands for — well, let me just say that LPBs don't get a lot of love from their slower connectivity comrades.)

WebRTC — and specifically WebRTC DataChannel (read more in Chapter 1) — represents a golden opportunity to increase the performance of multiplayer online games. WebRTC DataChannel provides a flexible and direct peer-to-peer channel for almost any kind of data, including game play data. This allows online gaming platforms to bypass the server and, along with it, any bottlenecks and inherent delays created as data is sent from one player to the server to another player and back again.

WebRTC DataChannel is designed for speed and low latency and can be configured in either reliable mode (for more complete and guaranteed in-order delivery) or unreliable mode (which is faster), depending on the needs of the game. Add to that the fact that games can be provided as web applications and played in any compliant browser with no downloads needed, and you've got — sorry for the gaming pun, but I really couldn't help myself — a winner.

Chapter 4

Where's the Money in WebRTC?

In This Chapter

- Licensing your way to revenues
- Selling tools for WebRTC
- Making money by selling WebRTC services
- Equipping WebRTC
- Improving the enterprise with WebRTC

The road to the future is littered with the skeletons of "the next big thing" that turned out to be more hype than hope. Web Real-Time Communications (WebRTC) has the potential to be a game changer in the communications industry, but before it gets the opportunity it will first need to show it can be a moneymaker. Because the bottom line is, after all, the bottom line.

In this chapter, I explore the different business cases for monetizing WebRTC, from software vendors to service providers.

Tapping into Licensing Revenues

Just because WebRTC is an open-source platform that anyone can use for free, it doesn't mean that WebRTC applications have to be free as well. In fact, many opportunities exist for application developers to profit from building WebRTC-based applications.

WebRTC is a framework that features a set of specifications for browser- or app-based, real-time communications; it isn't a set of finalized applications that can just be deployed on a server and turned on. The emerging WebRTC market represents a rich and untapped field for software vendors to create and license WebRTC applications for all kinds of communications purposes, from video chat services to medical applications — the sky is the limit here.

Selling Tools for Apps

Software vendors can also monetize WebRTC through application toolkits that enable software developers to embed WebRTC functionality into their existing apps or create their own WebRTC applications. These toolkits may include a set of software modules that simplifies and accelerates the application development process because many developers won't be WebRTC experts in the beginning. Toolkits may also feature the necessary Application Programming Interfaces (APIs), code snippets, and even customizable templates that allow developers to "skin" their WebRTC applications to match a specific appearance.

Toolkits can make it easier for developers to keep up with changing WebRTC standards as they continue to grow and take shape over time. In this case, the toolkit vendor acts as the WebRTC standards expert, while developers only need to keep their toolkit software up to date instead of keeping up with the latest WebRTC standards.

Vendors can also offer their WebRTC toolkits as a Platform as a Service (PaaS). This format gives developers access to an online toolkit as well as a hosting platform for the WebRTC applications they develop.

Making Money for Telcos and Other Service Providers

In addition to WebRTC toolkits and applications, companies may wish to offer soup-to-nuts WebRTC services, which may appeal primarily to companies that have already established

themselves in the hosted services market, such as hosted telecommunications or UC providers. This approach might take one of two paths:

- Integrating WebRTC functionality into an existing call center, videoconferencing, or other communications application that the provider offers
- Creating new WebRTC apps from scratch

Given the success of the Cloud model for many enterprise services, it's very possible that WebRTC as a Service may become a popular delivery method for creating and maintaining WebRTC applications. Certainly, there is an economy of scale/cost to be realized when enterprises can lease resources and capacity rather than build their own WebRTC application environment.

Building the Infrastructure

WebRTC — unlike many of the telecommunications technologies that came before — is less dependent on CapEx (or capital expenditures) investment to create new services. Deploying WebRTC requires some software, web servers (which may already be in place), and a gateway — not what could be characterized as high-ticket items in the traditional telco sense.

Still, opportunities do exist for network equipment vendors to generate revenue from WebRTC, especially among service providers that wish to integrate WebRTC into their existing networks. These opportunities include

- Installation of Session Border Controllers (SBCs) and WebRTC gateways
- Licensing for new codecs required by WebRTC
- Licensing and hardware upgrades to accommodate the rise in network traffic brought on by WebRTC applications

Benefiting Enterprise Users

As you may expect, certain cost benefits apply for enterprises that adopt WebRTC services. Otherwise, why would they even

bother, right? These benefits can be measured both directly and indirectly:

- **Direct benefits** include lower operational and capital expenses for communications. You don't incur a cost — not from upfront software licenses or ongoing support — for UC client software with WebRTC. Users simply use their existing apps or web browsers.
- **Indirect benefits** include the improvements in efficiency and customer service that WebRTC applications offer. Examples include the following:
  - Enhanced customer service opportunities that arise when customers can instantly connect to your enterprise via voice or video right from their web browsers
  - New customer interaction opportunities offered by WebRTC, such as telemedicine services that deliver anywhere/anytime service to patients
  - Instant, real-time collaboration that can truly take advantage of "anytime, anywhere, any device" communications

Chapter 5

Ten Things to Think About before You Deploy WebRTC

In This Chapter

- Establishing a business case
- Setting up the devices and picking the right browser
- Getting behind JavaScript
- Serving up the web part of WebRTC
- Keeping it secure with authentication
- Bringing it all together with transcoding

If you've been reading straight through this book, I hope you're excited about Web Real-Time Communications (WebRTC). But before you sell your Private Branch Exchange (PBX) for scrap metal and move everything to WebRTC, you need to make sure your business (including your network) is ready for WebRTC, too. In this chapter, I give you the ten most important things you should be thinking about *before* you deploy WebRTC.

The Business Case

Just like any technology investment, WebRTC won't magically make your business smarter or richer — for that, you need a business plan. As you align WebRTC with your strategic goals, keep in mind questions such as the following:

- What business problems can WebRTC solve for us today? Tomorrow?

- Does WebRTC address a communications need that we can't fill with our current solutions?
- Will using WebRTC help employees, partners, and customers communicate and collaborate better?
- How will WebRTC save us money and — more importantly — how will it make us money?

WebRTC is bigger than just voice and video. It can also be used for capturing images, sharing desktops, providing guidance in real time, exchanging files, and all sorts of other use cases.

The Sound and Vision

The majority of use cases for WebRTC in the near future center on voice and video communications. That said, who knows what uses people will invent for WebRTC in the future? And while you may not need to download software to use WebRTC, you do need a microphone (sound) and a camera (vision). Many devices, such as smartphones and tablets, already include these elements, but if you're planning to use WebRTC from your desktop, make sure you have the right equipment to join in on the conversation.

The Web Browser

One of the few things WebRTC does require is a native application or a compatible web browser. The World Wide Web Consortium (W3C) has defined the standards for WebRTC compliance to ensure that users can expect smooth and seamless communications from their browsers. Current versions of many of the leading browsers, including Google Chrome and Mozilla Firefox, support WebRTC today. Microsoft Internet Explorer supports a slightly different flavor of WebRTC while Apple's Safari browser supports WebRTC through third-party Application Programming Interfaces (APIs).

At the heart of WebRTC is a collection of APIs (getUserMedia and so on). An API provides an abstraction of some lower level capability to application programmers to make their lives easier (they don't need to know what's going on at the low level, just how to call the API). The authors of WebRTC had

to make certain judgment calls about how much complexity to abstract and how much to hide. A group of application and browser vendors argue that in some use cases more control is required and are authoring a new set of APIs in a standard called Object Real Time Communications (ORTC). Currently this is a separate initiative, but bit-by-bit much of this functionality is being merged with WebRTC, and it is widely expected that in a future release of the standard, WebRTC and ORTC will converge into one implementation with a range of APIs to suit every application developer.

Not sure if your browser supports WebRTC? You can always be assured of WebRTC compliance by downloading the latest version of Chrome, Firefox, or Opera.

JavaScript

JavaScript is a programming language used on hundreds of thousands of websites and, not surprisingly, in WebRTC communications too. When a website or app wants to enable a WebRTC session between users, it needs to send some instructions (the JavaScript) to the browser in order for the browser to know how to communicate with another user. This happens invisibly, in the background — the JavaScript is shared with the browser just as HTML, images, and other web content is. Ultimately, it's JavaScript that allows your website to bring up the WebRTC components in the browser, enabling real-time communication.

The Web Server

If you deploy WebRTC as a web service, you need a web server to run it on. In many cases, you won't even need to purchase a dedicated web server; you can simply add the WebRTC code and APIs to your existing web servers. Even so, careful consideration should be given to how and where in the website structure WebRTC is implemented and what use is made of it. There's more to enabling WebRTC on an existing website than just adding the WebRTC APIs and creating additional code on an HTML page. You also need to think about how the

WebRTC services will be used by your website. Here are some examples:

- Where in the site's navigation are the WebRTC services available or launched?
- How will the WebRTC session relate to the context of the web page?
- Is the session treated differently when you connect from different web pages?

Authentication

WebRTC sessions are secure by design, because the WebRTC standard calls for strong encryption mechanisms by default. In other words, you don't need to worry about third parties eavesdropping on your WebRTC communications (provided you haven't accidentally turned your volume level up to ten).

What you *might* want to worry about, however, is the identity of the person on the other end of your WebRTC conversation. That's the role of authentication. *Authentication* is using passwords or cryptographic tools to verify the identity of the parties involved in a conversation or session.

Note: I said that you *might* want to implement authentication. As you begin to deploy WebRTC services, decide if the service you're offering requires users to be authenticated or if it's preferable for them to be allowed to keep their anonymity. If you're a bank, for example, the answer is probably "yes" to using authentication. If you're setting up a customer service hotline for a consumer product, probably not.

The good news is that user authentication is an easily solved problem with many solutions already available. Almost any web or app authentication scheme can be used for WebRTC services. While that leaves a technical decision to make about which service to use, the more critical decision is a business one: Does your WebRTC application even require it?

Turn, Turn, Turn

One of the great advantages of the WebRTC standard is that it can direct traffic (voice, video, and data) directly between web browsers in a peer-to-peer fashion, bypassing any servers in the middle of the connection (a point I cover in more detail in Chapter 1). The advantage of this approach is that the data takes the most direct path, which ensures the best speed and performance possible. An added benefit is that a website owner avoids having to provide bandwidth capacity for all the sessions, which greatly reduces cost.

Due to technical limitations in some broadband environments, a web browser can't always establish a direct path on its own. Fortunately, WebRTC has some really smart work-arounds to this that include built-in procedures for intelligently selecting the best path without getting the user involved. These alternative paths are provided by a component called a *Traversal Using Relays around NAT (TURN) server.* TURN servers live in the data center alongside web servers, and they provide an alternative way to connect users when the preferred, direct path isn't possible.

You can consider TURN server capacity a necessity, but because WebRTC is so smart, it only uses TURN servers when absolutely necessary, which minimizes the capacity (and the cost) you need to expend.

The WebRTC Gateway

Sure, you could simply stick to browser-to-browser communications for WebRTC, but it can do so much more, too! For example, you may want to connect customers on your website with phone-based operators in your call center or connect smartphone users to a customer service representative through a special customer app.

To do this, you need more than WebRTC; you need a WebRTC gateway. A WebRTC gateway takes the WebRTC-based traffic and converts it into Session Initiation Protocol (SIP) to connect it to standard Voice over Internet Protocol/Unified

Communications (VoIP/UC) networks. This opens your closed WebRTC world up to the wider communications networks of the world, allowing you to connect WebRTC users to anyone in your network or anyone in the world.

Media Transcoding

As I talk about in Chapters 1 and 2, WebRTC uses several standard codecs for audio and video communications — codecs that may or may not align with the codecs used by your VoIP/UC network (flip back there now if you need a refresher). In the cases where the codecs are different, you need to translate between the two in order to enable WebRTC-to-VoIP/UC sessions — a job handled rather neatly by the Session Border Controller (SBC). I cover SBCs in Chapter 2.

The transcoding is usually done in an SBC. The SBC and WebRTC gateway work hand in hand, dealing with both signaling (connecting and disconnecting the calls) and the media (the voice and video streams themselves). Together the gateway and SBC provide interworking between disparate WebRTC and VoIP systems, ensuring calls can be seamlessly and successfully completed.

SIP Trunks

This is where WebRTC gets really sophisticated. What if you decide that you need to connect your WebRTC service beyond the confines of your VoIP network and out to the broader world? The most practical and economical way to do this is to commission a SIP trunk from your telephony provider.

For security and traffic control reasons, make sure you secure the SIP trunk connection at the border of your network by using an SBC. Fortunately, this can be the same SBC that provides transcoding and interworking with the WebRTC gateway between your WebRTC service and your VoIP network. With the addition of a SIP trunk, you complete the bridge between your WebRTC users and any other phone in the world, which is pretty cool when you think about it.

You could choose to connect your network to the outside world using a more traditional telephony services trunk such as an Integrated Services Digital Network (ISDN) trunk, but these connections are generally more expensive and require further specialized equipment. A SIP trunk is cheaper and easier.

Notes

Notes

Notes

Notes

Notes

Migrate your business applications to WebRTC!

Move beyond downloadable apps and browser plug-ins to WebRTC. This book helps you understand why WebRTC makes it easier for businesses to communicate with their customers and for co-workers and colleagues to communicate with each other — all without requiring downloads of client software or specially configured web browser setups.

- ***Simplify and enrich voice and video communications — deploy WebRTC***
- ***Grow your revenue base — increase the addressable market for carrier services with WebRTC***
- ***Transform legacy enterprise communication — migrate to WebRTC to increase customer satisfaction***
- ***Protect and integrate legacy enterprise networks with WebRTC — make sure you have deployed an SBC***

Open the book and find:

- **The features and benefits of WebRTC**
- **How WebRTC integrates into existing networks**
- **Interesting WebRTC services**
- **Who makes money with WebRTC**
- **How to leverage an SBC for WebRTC interworking and security**

Making Everything Easier!™

Go to Dummies.com®
for videos, step-by-step examples, how-to articles, or to shop!

FOR DUMMIES®
A Wiley Brand

Also available as an e-book

ISBN: 978-1-119-09878-2
Book not for resale