AVAYA

Advanced IP Communications
Application Development

Mastering Device, Media and Call Control

Using Avaya DMCC Dashboard

An essential tool for developing applications using the DMCC service of Avaya Communication Manager.

Advanced IP Communications
Application Development

Mastering Device, Media and Call Control

Using Avaya DMCC Dashboard

An Avaya DevConnect Publication

Mastering Device, Media and Call Control

© 2009 Avaya Inc. All Rights Reserved.
Avaya and the Avaya Logo are trademarks of Avaya Inc. and may be registered in certain jurisdictions.
All trademarks identified by ®, TM or SM are registered marks, trademarks, and service marks,
respectively, of Avaya Inc., with the exception of FORTUNE 500 which is a registered trademark
of Time Inc. All other trademarks are the property of their respective owners.
1/09 MIS4040-DEV

Contents

INTRODUCTION 1

Device, Media and Call Control in Intelligent Communications 1
Why You Should Read This Book 2
Why the DMCC Dashboard is Important to You 3
Product Releases 4
Assumed Knowledge 4
About the Avaya DevConnect Program 4
How this Book is Organized 5
Part I: DMCC Basics 5
Part II: Using the DMCC Dashboard 5
Icons Used in This Book 5

PART I: DMCC DASHBOARD BASICS 7

CHAPTER 1: PLATFORMS AND SERVICES 9

About Avaya Communication Manager 9
About Avaya Application Enablement Services 10
About the DMCC Service 11
First- and Third-party Call Control 12
Device Registration 12
Extensions and Devices 13
Supported Extension Types 13
Multiple Device Registration 13
The DMCC APIs and SDKs 14
Round-up 15

CHAPTER 2: INTRODUCING THE DMCC DASHBOARD 17

What is the DMCC Dashboard? 17
What Does the DMCC Dashboard Do and What is it Used For? 18
Who Should Use the DMCC Dashboard? 19
Summary of DMCC Features 20
Obtaining, Installing and Starting the DMCC Dashboard 21
Obtaining the DMCC Dashboard 21
Installing the DMCC Dashboard 21
Starting the DMCC Dashboard 21
DMCC Dashboard Thin-client Version 22
Accessing an AE Services and Communication Manager Installation . 22
Controllable Extensions 23
Round-up 24

Contents

CHAPTER 3: THE DMCC DASHBOARD USER INTERFACE 25
Layout of the DMCC Dashboard User Interface 26
Region 1: DMCC Service Controls 27
Region 2: DMCC Phone Controls 28
Region 3: XML Messages and Events 29
Region 4: Exceptions and Errors 30
Usability Features and Help 30
Intelligent Control Activation 30
Identification of Fields Related to Actions 31
Tooltips 32
On-line Help 32
Round-up 34

PART 2: USING THE DMCC DASHBOARD 35

CHAPTER 4: SESSIONS, MONITORS AND DEVICE REGISTRATION 37
The Main Tab 39
Quick Start Guide 40
Starting and Stopping Application Sessions 41
Starting Application Sessions 41
Keeping Sessions Alive 46
Starting Multiple Sessions 47
Getting Session IDs 47
Stopping Sessions 47
Getting First-party Device IDs 48
Listing Device IDs 50
Releasing Device IDs 50
Starting and Managing Event Monitors 51
Monitoring Events 51
Monitored Events 53
Managing Monitors 55
Viewing Active Monitors 56
Transferring Monitors 56
Stopping Monitors 56
Registering DMCC Devices 57
Registering a DMCC Device at an Extension 57
Media Control Modes 59
Registering DMCC Devices in Client Media Mode 60

Contents

Dependency Modes 61
Using Registered DMCC Devices 62
Unregistering DMCC Devices 62
Converting Telephone Number Formats 63
Converting E.164 Telephone Numbers 63
Converting Extension Numbers 64
Using the Dashboard IP Softphone 64
Controlling an Extension 64
Dashboard IP Softphone Options 65
Making a Call Using the Dashboard's IP Softphone 66
Sending Raw XML Messages to the DMCC Service 67
Configuring the DMCC Dashboard 69
Other Main Tab Functionality 69
Round-up 70

CHAPTER 5: THIRD-PARTY CALL CONTROL 71
Identifying Extensions, Calls and Connections 72
Third-party Device IDs 72
Call IDs 72
Connection IDs 73
The Call Control Tab 73
Preliminary Steps 74
Making and Answering Calls 75
Getting Third-party Device IDs 75
Making Calls 76
Getting Call IDs 78
Answering Calls 79
Third-party Call Control Options 80
Round-up 84

CHAPTER 6: SERVER-SIDE MEDIA CONTROL 85
About Server-Side Media Control 86
Server Media Mode RTP Media Parameters 86
Media Events 87
The Server Media Tab 88
Server-side Media Control Using Avaya IPCoDE 89
Preliminary Steps 89
Playing Prerecorded Messages 90
Server-side Call Recording 91

Contents

Dubbing Call Recordings 92
Detecting and Collecting DTMF Tones 92
Tone Detection 93
Tone Collection 93
Setting Tone Retrieval Criteria 93
Collecting DTMF Tones 94
Round-up 95

CHAPTER 7: CLIENT-SIDE MEDIA CONTROL 97
About Client-side Media Control 98
Setting RTP Media Parameters 98
Media Events 99
Media Stream Access Methods 100
The Client Media Tab 101
Client-side Media Control Using Avaya IPCoDE 102
Preliminary Steps 102
Client-side Media Control Example 103
Viewing RTP and RTCP Data 104
Redirecting Media Streams 105
Round-up 106

CHAPTER 8: FIRST-PARTY DEVICE CONTROL 107
About First-party Device Control 108
First-party Device Control in Applications 108
Getting Physical Element Information 108
Monitoring Phone Events 109
Activating Physical Elements 109
The Phone Commands Tab 110
Identifying Buttons 111
About Button Modules and Button Numbers 111
Deriving Button Identifiers 112
Preliminary Steps 113
Getting Information about Administered Buttons 114
Getting Information about a Single Button 115
Getting Information about all Buttons 117
Getting Information about Lamp States 117
Getting Other Information about Phone Sets 118
Controlling Phone Sets 119

Contents

Pushing Administered Buttons 119
Which Button to Press? 120
Registration and Security Code Options 121
Getting the Registration State 122
Validating and Changing Security Codes 122
Round-up 122

CHAPTER 9: GETTING ACTIVE CALL INFORMATION 123
About Call Information Links 123
The Link and Call Information Tab 124
Preliminary Steps 125
Getting Call Information 125
Getting Link Status Information 126
Round-up 127

CHAPTER 10: AUTOMATED TESTING AND PROTOTYPING 129
About Automated Testing and Prototyping 129
The Automated Testing Tab 130
Automated Testing Example 132
Getting XML Message Information 133
Rerunning Sequences of Actions 133
Rerunning Actions on Different Device IDs 134
Controlling Action Timings 136
Inserting Comments 137
Saving, Editing and Loading Scripts 137
Saving Sequences of Actions 137
Editing Scripts 138
Loading and Running Scripts 139
Round-up 140

CHAPTER 11: GET THE DASHBOARD AND GET STARTED! 141
Getting Started on Your Own 141
Key Take-aways and Top Ten DMCC Dashboard Facts 142

APPENDIX 145
DevConnect Developer and Partner Program 145
Application Enablement Services SDKs 148
Avaya IP Communications Development Environment 151

Figures and Tables

Figure 1-1: AE Services Schematic Diagram 10
Figure 2-1: The DMCC Dashboard User Interface 18
Figure 3-1: DMCC Dashboard Layout 26
Figure 3-2: DMCC Dashboard Control Tabs 27
Figure 3-3: The Simple DMCC Softphone Controls 28
Figure 3-4: XML Messages and Events Region 29
Figure 3-5: The Exceptions and Errors Region 30
Figure 3-6: Start Application Session Fields 31
Figure 3-7: Tooltip for the Reconnect Button 32
Figure 3-8: On-line Help for the Reconnect Button 32
Figure 3-9: Web-based Help for the Reconnect Method 33
Figure 4-1: The DMCC Dashboard Main Tab 39
Figure 4-2: Quick Start Options 40
Figure 4-3: Starting an Application Session 42
Figure 4-4: Start Application Session XML Messages 44
Figure 4-5: Session IDs List Box 46
Figure 4-6: Getting Device IDs 48
Figure 4-7: Device IDs List Box 49
Figure 4-8: Event Registration Tabs 52
Figure 4-9: Get Device ID List Event 55
Figure 4-10: Monitor IDs List Box 55
Figure 4-11: Transferring Monitors 56
Figure 4-12: Stopping Monitors 57
Figure 4-13: Registering DMCC Devices 58
Figure 4-14: Registering DMCC Devices in Client Media Mode 60
Figure 4-15: Converting E.164 Telephone Numbers 63
Figure 4-16: Selecting an Extension to Control 65
Figure 4-17: The DCP Phone at Extension 40010 66
Figure 4-18: DMCC Dashboard IP Softphone on Call 67
Figure 4-19: The DCP Phone at Extension 40011 in the Ringing State ... 67
Figure 4-20: Sending XML Messages Direct to the DMCC Service 68
Figure 5-1: The Call Control Tab 73
Figure 5-2: Getting Third-party Device IDs 75
Figure 5-3: Getting Third-party Device ID Results 76
Figure 5-4: Making a Call 76

Figure 5-5: Answering a Call 79
Figure 6-1: The Server Media Tab 88
Figure 6-2: Playing Messages..... 90
Figure 6-3: Starting Server-side Call Recording..... 91
Figure 6-4: Starting Call Recording Dubbing 92
Figure 6-5: Setting Tone Retrieval Criteria..... 94
Figure 7-1: The Client Media Tab 101
Figure 7-2: Displaying RTP Data 105
Figure 8-1: The Phone Commands Tab..... 110
Figure 8-2: Getting Button Information 115
Figure 8-3: Pushing an Administered Button 120
Figure 8-4: Registration and Security Code Options..... 121
Figure 9-1: The Link and Call Information Tab 124
Figure 10-1: The Automated Testing Tab 130
Figure 10-2: Viewing Command Error Details..... 131
Figure 10-3: The Example Button Pushes 132
Figure 10-4: Viewing XML Message Details 133
Figure 10-5: Rerunning a Series of Commands..... 134
Figure 10-6: Automated Button Presses 134
Figure 10-7: Viewing Action Device IDs 135
Figure 10-8: Inserting a Delay Between Actions 136
Figure 10-9: Saving a Sequence of Actions to a Script 138
Figure 10-10: Editing a Script File 139
Figure 10-11: Running a Script 140

TABLES

Table 1-1: Summary of DMCC Dashboard Features 20
Table 4-1: Monitored Events 53
Table 5-1: Third-party Call Control Options 82
Table 5-2: Snapshot Options 83
Table 5-3: Logical Device Options 83
Table 5-4: Other Call Control Tab Options 84

Introduction

Device, Media and Call Control in Intelligent Communications

Businesses that want to be both agile and competitive in today's complex marketplace need to maximize their use of the latest communications capabilities—IP Telephony, Unified Communications, Presence, advanced Contact Center solutions, and much more. As a result, application developers are increasingly being asked to integrate these communications capabilities directly into business processes—for example in enterprise portals, desktop applications or even back office systems.

Avaya calls this "Intelligent Communications"—the ability to seamlessly integrate communications capabilities into the fabric of a business in ways that profoundly transform how it operates at all levels.

Intelligent Communications is about much more than simplifying access to communications from any device or application. It is also about managing and synchronizing the wide range of data and information that makes communications more effective—contact and profile information, presence and availability status, even personal preferences and business priorities—and using that information in real time to coordinate business activities.

This book will help you gain greater insight into the power and flexibility of intelligent communications offered by Avaya through Application Enablement (AE) Services, its Device, Media and Call Control (DMCC) service and—the primary focus of this book—the DMCC Dashboard:

- **Application Enablement (AE) Services** is the software layer of Avaya Communication Manager (Avaya's flagship IP telephony software platform) that exposes and abstracts its communications capabilities, making it possible to integrate communication services across a business.
- **AE Services Device, Media and Call Control (DMCC)** service is the software capability that enables client applications to leverage Communication Manager's physical device, media, and first- and third-party call control capabilities.

- **DMCC Dashboard** is a GUI-based client application that allows application developers to exercise and observe just about all of the device, media and third-party control capabilities supported by Communication Manager and exposed via the AE Services DMCC service. The DMCC Dashboard is an ideal way to learn about and demonstrate the capabilities of the DMCC service while also getting access to tools that will help you create your own runtime IP communications applications.

Available at no charge to all registered Avaya DevConnect members, the DMCC Dashboard gives you the insight, understanding and tools you need to simplify and speed the whole application development process, from learning about the APIs through to testing and debugging.

Why You Should Read This Book

Whether you're involved in IP communications application development, or someone who just wants to know more about IP communications applications and what they can do, you'll find this book immensely valuable.

By learning about the device, media and call control capabilities provided by Communication Manager (and exposed via AE Services DMCC service) and the features, benefits and use of the DMCC Dashboard, you'll also gain an appreciation of the capabilities of the underlying communications management platform, a better understanding of how the functionality of the DMCC service is invoked and the effects of combining individual capabilities.

By mastering device, media, and call control APIs exposed by AE Services, you will move beyond thinking of communications only in its most basic form of "phone calls." Instead, you will recognize that the communications capabilities accessible from a desk phone—from advanced routing to presence, even call recording—can just as easily be part of your enterprise portal, part of your workflow engines, even part of your business processes. With this insight, you'll be capable of delivering intelligent communications that transform ordinary businesses into extraordinary ones.

Why the DMCC Dashboard is Important to You

The DMCC Dashboard is ideal for learning about and demonstrating the capabilities of the DMCC service. The DMCC Dashboard also has features designed to help developers create their own runtime IP communications applications. In particular, the DMCC Dashboard allows you to:

- View the XML messages exchanged between the DMCC Dashboard, acting as a client application, and the DMCC service. These messages can be used as templates for the XML that your own client applications need to generate and handle.
- Send XML messages directly to the DMCC service and observe the results. This feature allows you to unit test XML messages created for your own client applications and to observe the effects of making fine changes to requests.
- Monitor events at extensions managed by Communication Manager. Observing generated events allows you to understand the effects of actions and how the corresponding callback methods can be implemented in your own client applications.
- Prototype simple applications by stepping through the required method calls in the DMCC Dashboard.
- Automate testing activities by saving DMCC Dashboard operations to a script that can subsequently be replayed.

So, as well as being a great demonstration tool, the DMCC Dashboard can be used by developers to:

- Learn about the device, media and third-party call control functionality that is available to them for inclusion in their applications.
- Help design and prototype applications by observing the effects of combining individual DMCC service method calls.
- Aid in the testing and troubleshooting of applications under development.

Product Releases

This book is based on the releases of products that were current at the time of writing, namely:

- Avaya Communication Manager release 5.1
- Avaya Application Enablement Services release 4.2
- DMCC Dashboard release 4.2

Features due to be included in future releases of the DMCC Dashboard are described in the book, indicated by the "Coming Soon" icon. Note, however, that the implementation of these new features is not guaranteed.

Assumed Knowledge

To get the most out of this book it would be useful to have a good understanding of IP telephony and computer telephony integration. It is also advantageous, but not essential, to have a working knowledge of Avaya Communication Manager and Avaya Application Enablement (AE) Services.

About the Avaya DevConnect Program

DevConnect is Avaya's developer and partner program, open to IP communications application developers, Systems Integrators, ISV's, IHV's and customers alike. Basic membership is free, offering no-charge access to technical education, product API documentation, Software Development Kits (SDKs), sample applications, technical support and more.

Among the developer tools available to DevConnect members is the Avaya IP Communications Development Environment (IPCoDE), which can be used in conjunction with the DMCC Dashboard to simulate a complete IP Communications environment for both SIP and Avaya H.323-based solutions.

Companies that desire an expanded relationship with Avaya can apply to become an enhanced member, which offers greater technical support, discounted procurement for lab systems, and co-marketing benefits based on proven compliance with Avaya solutions.

For more information, and to start with a free registered-level membership, visit www.avaya.com/devconnect.

How this Book is Organized

This book is divided into two parts:

Part I: DMCC Basics

In Part I you will learn about the platforms and services that DMCC Dashboard uses. You will gain an understanding of what the DMCC Dashboard is, what it's used for and who should use it. You will also gain an appreciation of the value of the DMCC Dashboard, both to developers who want to make use of the DMCC service in their applications, and to other users who want to learn about the capabilities of Avaya Communication Manager exposed via the DMCC service.

Part II: Using the DMCC Dashboard

In Part II you will learn how to use the DMCC Dashboard to exercise the capabilities of the DMCC service, including physical device control, client- and server-side media control and basic third-party-call control. You will also discover how the DMCC Dashboard can be used to prototype and test applications under development. The chapters in this part are organized based on the tabs of the DMCC Dashboard user interface.

Icons Used in This Book

Coming Soon: information about DMCC Dashboard functionality due to be included in post 4.2 releases

Important information about the DMCC Dashboard

Additional information for developers

Helpful advice for DMCC Dashboard users

Part I:
DMCC Dashboard Basics

What's in Part I?

This part of the book gives you an introduction to, and overview of, the DMCC Dashboard.

In Part I you will learn about the platforms (Avaya Communication Manager and Avaya Application Enablement (AE) Services) and services (AE Services Device, Media, and Call Control (DMCC) service) on which the DMCC Dashboard runs. You will gain an understanding of what the DMCC Dashboard is, what it's used for and who should use it. You will also gain an appreciation of the value of the DMCC Dashboard, both to developers who want to make use of the DMCC service in their applications, and to general users who want to learn about the capabilities of the DMCC service.

This part also introduces you to the DMCC Dashboard's user interface, including its layout and other features that make it easy and intuitive to use.

By the end of Part I you'll have the knowledge you need to go on to explore the DMCC Dashboard in much greater depth and learn the details of how it is used, in Part II.

8

Chapter 1
Platforms and Services

In this chapter:

- Introducing Avaya Communication Manager
- Enabling client application development using Avaya Application Enablement Services
- Exploring the capabilities of the AE Services DMCC service
- Understanding the role of the DMCC Dashboard

Before looking at the DMCC Dashboard in detail, it's useful to understand something of the underlying platforms and services. In this chapter, you'll find an overview of:

- **Avaya Communication Manager:** the platform which supports the underlying call control capabilities exercised by the DMCC Dashboard.
- **Avaya Application Enablement (AE) Services,** which provides connectivity between client applications and Avaya Communication Manager.
- **The AE Services Device, Media, and Call Control** service used by the DMCC Dashboard.

About Avaya Communication Manager

Avaya Communication Manager is Avaya's flagship IP telephony software platform. It contains robust call processing capabilities, advanced workforce productivity and mobility features, built-in conferencing and contact center applications, and support for a variety of wired and wireless end-user communications devices. Avaya Communication Manager scales from as few as 100 users on a single system up to more than one million devices on a single network. You'll find more information about Communication Manager on the Avaya web site (http://www.avaya.com).

Avaya Communication Manager exposes a series of low-level, proprietary application programming interfaces (APIs). These APIs allow other software products to work with Avaya Communication Manager. With the addition of Avaya Application Enablement Services, the APIs also allow software developers to create their own applications that interact with, and leverage the capabilities of, Avaya Communication Manager.

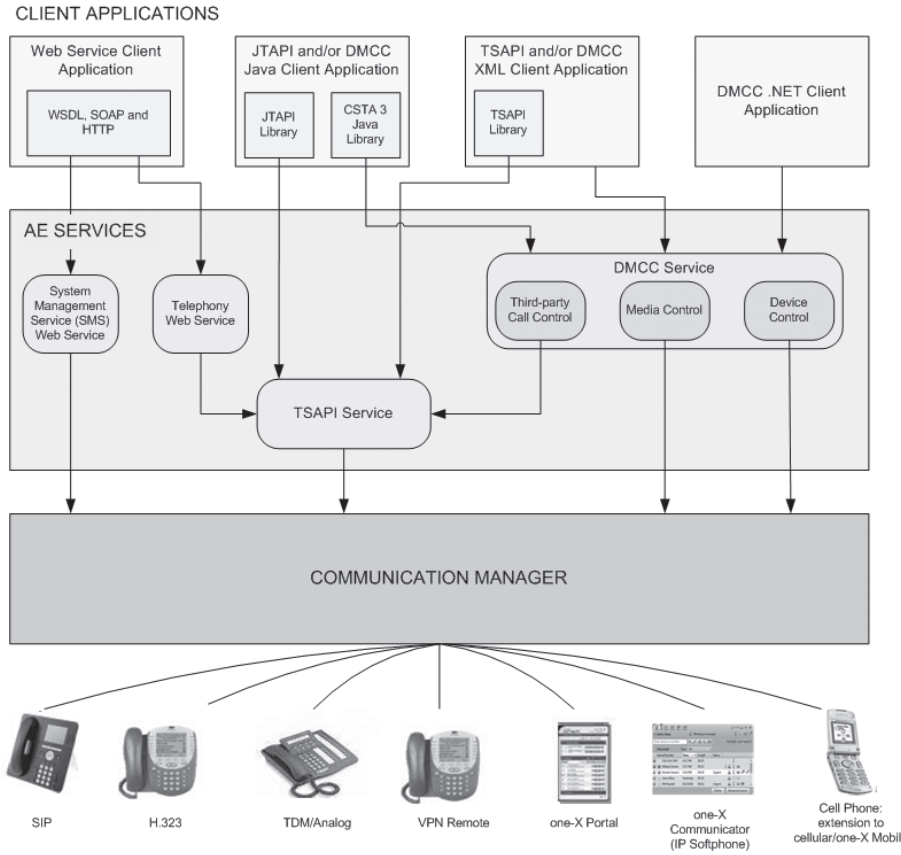

Figure 1-1: AE Services Schematic Diagram

About Avaya Application Enablement Services

Avaya Application Enablement (AE) Services is a server-based software application that provides connectivity between client applications and Avaya Communication Manager. AE Services incorporates a number of discrete services that provide high-level abstractions of the proprietary

Communication Manager APIs. Using AE Services, software developers can create client applications written in the programming language or protocol of their choice.

Figure 1-1 is a schematic diagram of the primary services available for application enablement and the types of client applications that use them.

About the DMCC Service

This book is about the DMCC Dashboard, so it's no surprise that it is the DMCC service that we are most concerned with here.

DMCC stands for "Device, Media, and Call Control". As the name suggests, the AE Services DMCC service enables access to Communication Manager's device, media and third-party call control functionality:

- **Device contro**l allows applications to manipulate and monitor the physical aspects of devices managed by Communication Manager, such as buttons, lamps, the display and the ringer. Applications can simulate manual actions on devices and obtain the status of their physical elements.
- **Media control** allows applications to access voice stream RTP data for the purposes of recording or analysis, and to send RTP data as outgoing voice streams. The AE Services server can record and playback WAV format files when it is managing the media stream for a device.
- **Call control** allows applications to perform basic third-party call control, such as placing calls, creating conference calls, deflecting calls, reconnecting calls, and monitoring call control events.

In addition, the DMCC service provides:

- **Logical Device Services** which enable applications to use call forwarding and "do not disturb" capabilities.
- **Snapshot Services** which allow applications to obtain information about the devices participating in calls.
- **Monitoring Services** which allow applications to request asynchronous notification of events.

First- and Third-party Call Control

The device and media control provided by the DMCC service each use what is known as "first-party call control". The call control part of DMCC actually refers to "third-party call control": so what's the difference between these call control methods?

First-party Call Control allows applications to take control and monitor physical devices involved in calls. So, for example, to place two parties in a call the application takes control of the physical device at the calling party's extension, and invokes methods to emulate taking the device off hook and pressing the appropriate keys to dial the digits that comprise a called party's telephone number. This gives an application fine-grained control of, and information about, an endpoint's state.

Third-party Call Control allows applications to control calls remotely. Thus, to place two parties in a call, the application simply issues a high-level instruction that causes both parties' extensions to be dialed and connected into a call.

Device Registration

To be able to perform first-party device and media control at an extension, a client application (such as the DMCC Dashboard) must use the AE Services DMCC service to register itself as a "DMCC device" on Communication Manager.

When registered as a DMCC device, the client application can act as a logical IP softphone to control and monitor physical aspects of the extension (button pushes, lamps, the display, etc.), or access and control the media streams at the extension.

Client applications do not need to register themselves at extensions to perform third-party call control.

Extensions and Devices

What is an extension and how does it differ from a device? In this book, an extension is defined as a Communication Manager station that has been provisioned with an extension number so that calls can be made to and from it. A device is a physical, virtual or logical entity that allows you to use and control that extension, such as a traditional physical set, an IP softphone or a client application. You can think of the relationship of a device to an extension as being like the relationship of an electrical appliance to the wall socket it is plugged into. As you will learn later, it's possible to have up to three devices at an extension: the analogy would be to attach a three-way adaptor to the wall socket so that three appliances can be plugged into it.

Supported Extension Types

Client applications can only register themselves as DMCC devices against extensions that have been IP softphone-enabled on Communication Manager. It is only possible to softphone-enable extensions that are administered as:

- DCP
- Avaya H.323 IP softphones

System Administrators can softphone-enable extensions at the Communication Manager's Stations screen.

Note: It is not possible to register a DMCC device against a SIP extension.

Multiple Device Registration

Using AE Service release 4.1 or higher, it is possible to register up to three devices against a common extension; in earlier releases, only one device could be registered.

Where multiple device registration is supported, the number of DMCC devices that can be registered against an extension is determined as follows:

- If there is no physical set or Avaya IP softphone registered at the extension, the client application can register up to three DMCC devices.
- If there is a physical set or Avaya IP softphone registered at an extension, the client application can register up to two DMCC devices.
- If a physical set and Avaya IP softphone share control of an extension, the client application can register one DMCC device.

The registration process includes defining dependency and media modes for the DMCC device, which in turn define the control capabilities available to the application. Certain media modes also require the DMCC device's codecs and media encryption method to be defined.

Device registration is fundamental to understanding the DMCC service and operating the DMCC Dashboard. You'll learn more about registration and the various mode settings in *Chapter 4*.

The DMCC APIs and SDKs

Depending on the programming language used, applications interface with the DMCC service using one of the following access methods:

- **DMCC Java API:** used by applications written in the Java programming language
- **DMCC .NET API:** used by applications written in the C#, Visual Basic or Visual C++ .NET programming languages
- **DMCC XML protocol description:** used by applications written in any programming language that supports the sending and receiving of XML data over a network connection. Applications that use this access method are typically written in C or C++.

The DMCC .NET API can present itself as an ActiveX control. This means you can write web pages for Internet Explorer that use DMCC capabilities. The DMCC .NET SDK includes a softphone written in DHTML to give you an idea of how powerful this can be.

Each of these access methods is an implementation of the CSTA III (Computer-Supported Telecommunications Applications III) standard (ECMA 269), plus Avaya-specific extensions, which allow access to a common set of Communication Manager capabilities.

The access methods are based on a client/server model, where the application is the client, and where AE Services and Communication Manager act together as the server. Thus, the DMCC service access methods allow applications to:

- request services of Communication Manager
- request notification of asynchronous events on Communication Manager

Each access method has its own SDK, available to registered members as a no-charge download from the DevConnect web portal (http://www.avaya.com/devconnect). Each DMCC SDK provides the tools developers need to create applications that use the DMCC service, including sample applications, API documentation, client-side libraries and so on.

Round-up

In this chapter you were introduced to Avaya Communication Manager, Avaya Application Enablement Services, the DMCC service and its APIs, and learned how these relate to one another. So the question now is, how does the DMCC Dashboard fit into all of this?

Well, the DMCC Dashboard is actually a client application that interfaces with the DMCC service and exercises nearly all of its capabilities. The DMCC Dashboard's user-friendly interface makes it easy for developers and other interested parties to try out and learn about the available functionality.

But that's not all: the DMCC Dashboard can be used to help developers create, prototype and test their own DMCC applications. The DMCC Dashboard is an essential tool for anyone wanting to learn about or develop applications against the DMCC service, as you will see in the next chapter.

Chapter 2

Introducing the DMCC Dashboard

In this chapter:

- Finding out what the DMCC Dashboard is and what it looks like
- Investigating what the DMCC Dashboard is and who it is intended for
- Obtaining the DMCC Dashboard
- Installing the DMCC Dashboard and setting it up ready for use

In *Chapter 1*, you learned about the platform and services that provide the underlying capabilities that the DMCC Dashboard exercises, namely Avaya Communication Manager, Avaya Application Enablement Services in general, and the DMCC service in particular. In this chapter, you will be introduced to the DMCC Dashboard itself: what it is, what it does, and how it can be used by software developers and others to help them understand the capabilities of the DMCC service and create their own applications that leverage those capabilities. You'll also discover how you can obtain and install the DMCC Dashboard on your desktop machine. Finally, you'll be advised on various options for accessing an installation of Avaya Application Enablement Services and Avaya Communication Manager to run the DMCC Dashboard against.

What is the DMCC Dashboard?

The DMCC Dashboard is essentially a client application that interfaces with the DMCC service and exercises just about all of its capabilities. The DMCC Dashboard happens to have been written in C# .NET using the DMCC .NET SDK, but the capabilities and concepts it demonstrates apply equally to all of the programming languages supported by the DMCC service access methods and SDKs.

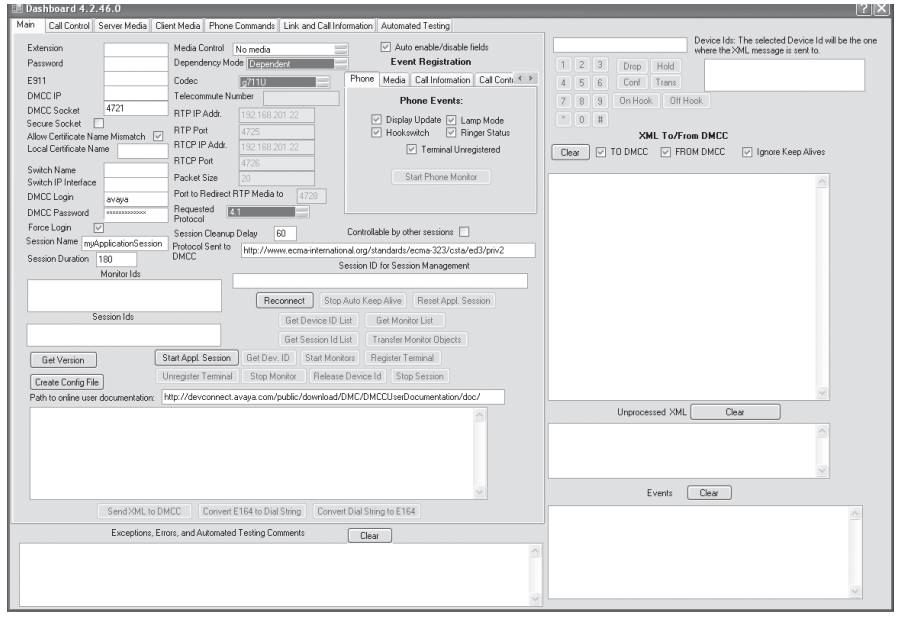

Figure 2-1: DMCC Dashboard when it is first started

In *Chapter 3* you'll explore the main elements of the DMCC Dashboard layout and in later chapters you'll examine the user interface in much closer detail.

What Does the DMCC Dashboard Do and What is it Used For?

The primary purpose of the DMCC Dashboard is to allow users to exercise and observe just about all of the capabilities of the DMCC service. The intuitive, user-friendly, graphical interface makes it easy for users to appreciate the functionality supported by the DMCC service.

Thus, the DMCC Dashboard is a great tool for anyone who wants to learn about the capabilities of the DMCC service, or demonstrate them to others. But the DMCC Dashboard also has features that can be used to support the creation of real-life applications that include DMCC functionality. For example, the DMCC Dashboard can be used to:

- Monitor the XML messages exchanged between the DMCC Dashboard and the DMCC service. Developers do not need to use a network sniffer, such as Ethereal, or access the AE Services logs to monitor XML messages.
- Monitor events at extensions connected to the same Avaya Communication Manager and AE Services servers.
- Send XML messages directly to the DMCC service and observe the results.
- Prototype simple applications by stepping through the required method calls in the DMCC Dashboard, and using the resultant XML messages and events to include the functionality in another application. Developers using the DMCC XML access method can copy XML directly from the DMCC Dashboard and use it as a template for the messages they want to send from their own applications.
- Automate testing activities by saving DMCC Dashboard operations to a script that can subsequently be replayed. DMCC Dashboard test scripts could be used in network tests or for regression testing applications.

Who Should Use the DMCC Dashboard?

The DMCC Dashboard is primarily targeted at software developers who wish to use the DMCC service to include device, media and/or basic third-party call control in their applications. But as you have already seen, it is a valuable tool for anyone who wants to have a better appreciation of the functionality that DMCC service is capable of providing.

Remember, although the DMCC Dashboard is a .NET application it is of equal value to developers using any of the DMCC service access methods.

Table 1-1
Summary of DMCC Dashboard Features

The DMCC Dashboard release 4.2 ...

- Exposes the functionality of the DMCC service in a single application.
- Allows users to exercise nearly all of the device, media and third-party control features of the DMCC service.
- Supports the following third-party call control options:

Alternate Call	Make Call
Answer Call	Reconnect Call
Clear Call	Retrieve Call
Clear Connection	Single Step Conference Call
Conference Call	Single Step Transfer Call
Consultation Call	Snapshot Call
Deflect Call	Get Do Not Disturb
Generate Digits	Get Forwarding
Get Third Party Device ID	Set Display
Hold Call	Set Do Not Disturb

- Includes tool tips that provide quick information on most controls.
- Provides context-sensitive links to detailed information on many controls.
- Provides the ability to record actions and save them to a script that can then be loaded and executed on the DMCC Dashboard to facilitate automated testing.
- Provides the ability to send any XML message to the DMCC service. This ability is particularly valuable to C programmers using the raw XML interface to the DMCC service.
- Provides the ability to monitor all XML messages going to and from the DMCC service.
- Can receive raw RTP/RTCP data and redirect it to another IP address.
- Can set monitors to notify developers when particular events occur. The types of events that can be notified range from lamp and display updates to the delivery of RTP data to the DMCC Dashboard.
- Allows users to view XML messages that arrive during a specified time period.
- Supports the transfer of devices and monitors from one session to another.
- Can simultaneously communicate with multiple DMCC services and with multiple devices

....and in future releases, the DMCC Dashboard...

- Saves incoming RTP data to a file that can be played back on the user's PC.
- Allows users to highlight new features.
- Provides the ability to inject XML messages into the DMCC Dashboard, as though they came from DMCC service, and observe the results.

Obtaining, Installing and Starting the DMCC Dashboard

Obtaining the DMCC Dashboard

The DMCC Dashboard is available to registered members as a no-charge download from the DevConnect website. To access the download:

1. Go to: http://www.avaya.com/devconnect
2. If you are already a DevConnect member, select **Member Login** and enter your credentials. Otherwise, select **Not a Member?** and sign-up for free registered membership.
3. Navigate to the DMCC Dashboard topic: **Quick Links: Products and SDKs -> Application Enablement Services –> AE Services: DMCC Dashboard.**
4. Follow the instruction under "How to Obtain the DMCC Dashboard".

Installing the DMCC Dashboard

To install the DMCC Dashboard, simply extract the downloaded ZIP file to a directory on your desktop machine.

Starting the DMCC Dashboard

To start the DMCC Dashboard, run the `dashboard.exe` file.

The `ServiceProvider.dll` file must be located in the same directory as `dashboard.exe`.

To view the whole dashboard on your computer screen, set the resolution to at least 1152 x 800.

DMCC Dashboard Thin-client Version

In addition to the Windows-based, thick-client version of the DMCC Dashboard described above, an Internet Explorer-based, thin-client version is also included in the download file. Both variants have similar features and exercise similar capabilities of the DMCC service. The thin-client version demonstrates how the .NET `SeviceProvider.dll` *component can be used in a browser. Each option has an Example code link to sample JavaScript code showing how the corresponding interface can be used in a browser-based client application.*

To start the thin-client web version, open `dashboard.html` *in Internet Explorer.*

***Note:** Before you can use the thin client version, the ServiceProvider.dll file must be placed on a web server and the dashboard.html file must be edited to point to the location of the serviceProvider.dll. There are comments in the* `dashboard.html` *file which provide further guidance.*

Accessing an AE Services and Communication Manager Installation

Before you can use the DMCC Dashboard, you need access to an installation of Avaya Communication Manager and Avaya Application Enablement Services to run it against. There are a number of options for obtaining access:

- If you have an operational installation of AE Services and Communication Manager on your company network, you may be able to use that. Higher-level Gold and Platinum DevConnect members may be eligible to procure discounted systems for development purposes as part of their membership benefits.
- All Registered-level DevConnect members and above can book a session in the AE Services and Communication Manager Remote Lab. Details are available on the DevConnect website under **Developer Resources: Remote Labs.**

- Purchase a copy of the Avaya IP Communications Development Environment (Avaya IPCoDE). Avaya IPCoDE includes VMWare images of AE Services and Communication Manager that can be installed and run on a desktop machine. Information about obtaining and installing Avaya IPCoDE is available on the DevConnect website under **Quick Links: Products & SDKs –> IP Communications Development Environment.**

The examples and screenshots used in this book are based on running the DMCC Dashboard against Avaya IPCoDE, installed on the same desktop machine.

Controllable Extensions

In addition to an installation of Communication Manager and AE Services, the DMCC Dashboard needs some extensions to monitor and control. The extensions must be at stations managed by Communication Manager.

Avaya IPCoDE is pre-provisioned with three IP softphone and three DCP extensions that can be controlled and monitored by the DMCC Dashboard:

- IP softphone extensions: 32129, 32130, and 32131
- DCP extensions: 40001, 40002, and 40003

The Avaya IPCoDE DCP extensions are provisioned with virtual DCP telephones that can be controlled by the DMCC Dashboard. However, the DCP extensions do not support audio, so they do not generate the media streams needed to exercise call recording, etc. If you want to use the DMCC Dashboard to exercise the media control capabilities, you can install an Avaya IP Softphone on your development machine and log in to one of the IP softphone extensions.

The examples in this book use the three DCP extensions, together with an Avaya IP Softphone (release 5.2) logged into extension 32129. You can download an Avaya IP Softphone from the Avaya support website (http://support.avaya.com).

Round-up

In this chapter you were introduced to the DMCC Dashboard and its many possible uses. Hopefully, you now have an appreciation of the value the DMCC Dashboard provides, not just to developers who want to include DMCC functionality in their applications, but also to anyone who wants to learn more about the capabilities of the DMCC service.

Before going on to explore the DMCC Dashboard in much greater detail and learning how it is used, it's worth finding out about the layout of the user interface, its usability features and help options, all of which are described in the next chapter.

Chapter 3

The DMCC Dashboard User Interface

In this chapter:

- Exploring the layout of the DMCC Dashboard user interface
- Looking at the features of the DMCC Dashboard that make it easier to use
- Finding out about the various levels of help available and how to access them

In the previous chapter you were introduced to the DMCC Dashboard and were given an overview of what it is and what it can be used for. In this chapter you are going to explore general aspects of the user interface, including its layout, features and help options.

Layout of the DMCC Dashboard User Interface

Figure 3-1 shows the four main regions of the DMCC Dashboard. These are:

Region 1: DMCC service controls

Region 2: DMCC phone controls

Region 3: Messages and events

Region 4: Exceptions and errors

Let's look at each region in turn.

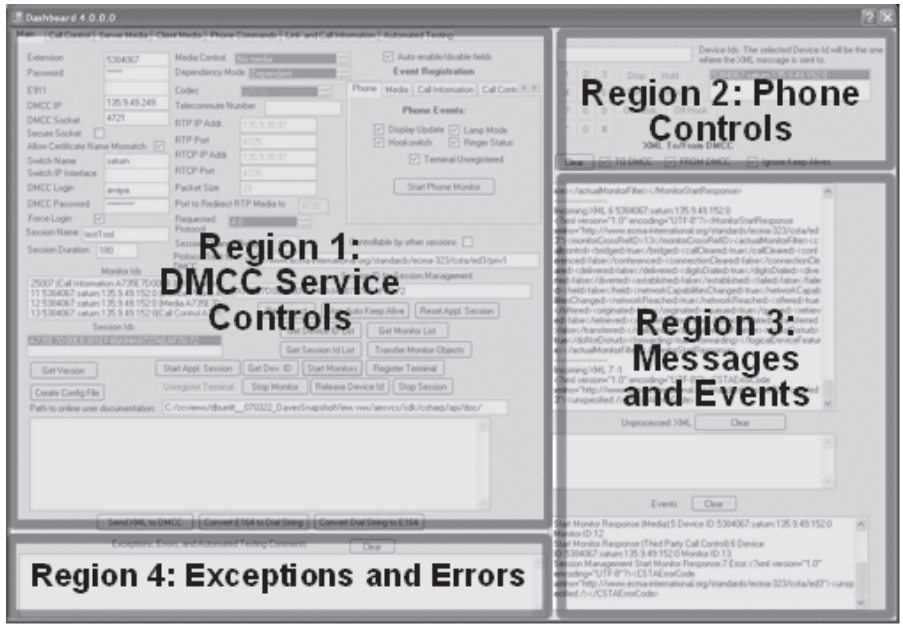

Figure 3-1: DMCC Dashboard Layout

Region 1: DMCC Service Controls

This area of the DMCC Dashboard comprises a series of tabs that group sets of related controls and allow users to access DMCC service functionality.

From left to right, the tabs are:

- **Main:** used to start and maintain sessions with the DMCC service, implement monitors and register devices at extensions for first-party device control.
- **Call Control:** used to exercise third-party call control functionality.
- **Server Media:** used for server media control. When a device is registered in Server Media Mode, its media is delivered to the DMCC server.
- **Client Media:** used for client media control. When a device is registered in Client Media Mode, its media is delivered to a user-specified IP address.
- **Phone Commands:** used for advanced first-party control of a registered device, including pushing buttons, getting lamp statuses and getting phone display contents.
- **Link and Call Information:** used to get information about calls and links between the DMCC and Communication Manager servers.
- **Automated Testing:** used to save, edit, load and run series of DMCC Dashboard button pushes, for the purposes of automated testing.

In the chapters that follow, you'll look at each of these tabs in detail to learn more about how the DMCC Dashboard exercises the capabilities of the DMCC service.

Figure 3-2: DMCC Dashboard Control Tabs

Region 2: The Simple DMCC Softphone Controls

This comprises the controls used for simple first-party control of extensions against which the DMCC Dashboard has registered itself as a DMCC device.

The simple softphone comprises the following fields and controls:

- **Registered Devices List:** The list box on the right-hand side of the softphone lists the Device IDs associated with extensions at which the DMCC Dashboard has registered as a DMCC device. The Device ID of the extension currently controlled by the softphone is highlighted. When softphone or other controls are selected on the DMCC Dashboard, the generated XML messages are sent to the AE Services DMCC service.
- **Controlled Device Display:** The text box at the top of the softphone shows the message currently displayed in the top line of the physical set or IP softphone at the extension against which the selected DMCC device is registered. The field typically displays Caller ID information.
- **Keypad:** The keypad is used to send standard telephone key press commands to the AE Services DMCC service to control the extension against which the selected DMCC device is registered.
- **Phone Command:** These buttons are used to send simple commands to the AE Services DMCC service to control the extension against which the selected DMCC device is registered. Users can emulate taking the controlled extension off hook, put it back on hook and drop, hold, transfer or conference calls at that extension.

Note: Access to additional first-party device control commands, such as customizable softkeys and speed dial buttons, is available on the **Phone Commands** tab in the DMCC service controls region.

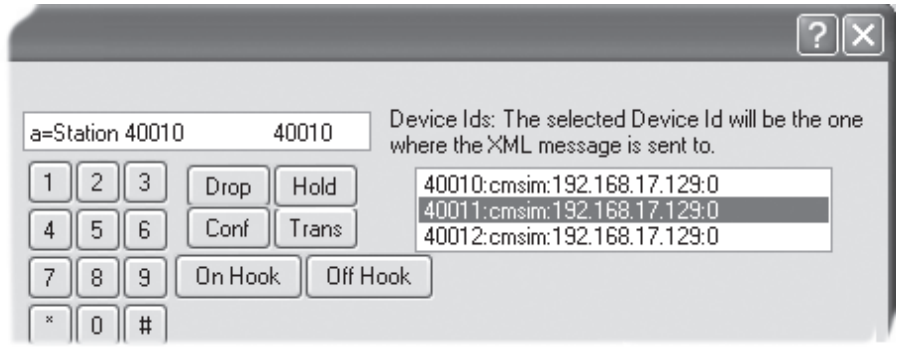

Figure 3-3: The Simple DMCC Softphone Controls

Region 3: XML Messages & Events

The XML Messages and Events region contains scrollable text boxes that display the XML messages sent between the dashboard and the AE Services DMCC service, unprocessed XML, and details of monitored events at the controlled device. Users can optionally select to view just XML messages sent to the DMCC service, just messages received from the DMCC service, or to ignore XML messages that are sent to keep the current session alive.

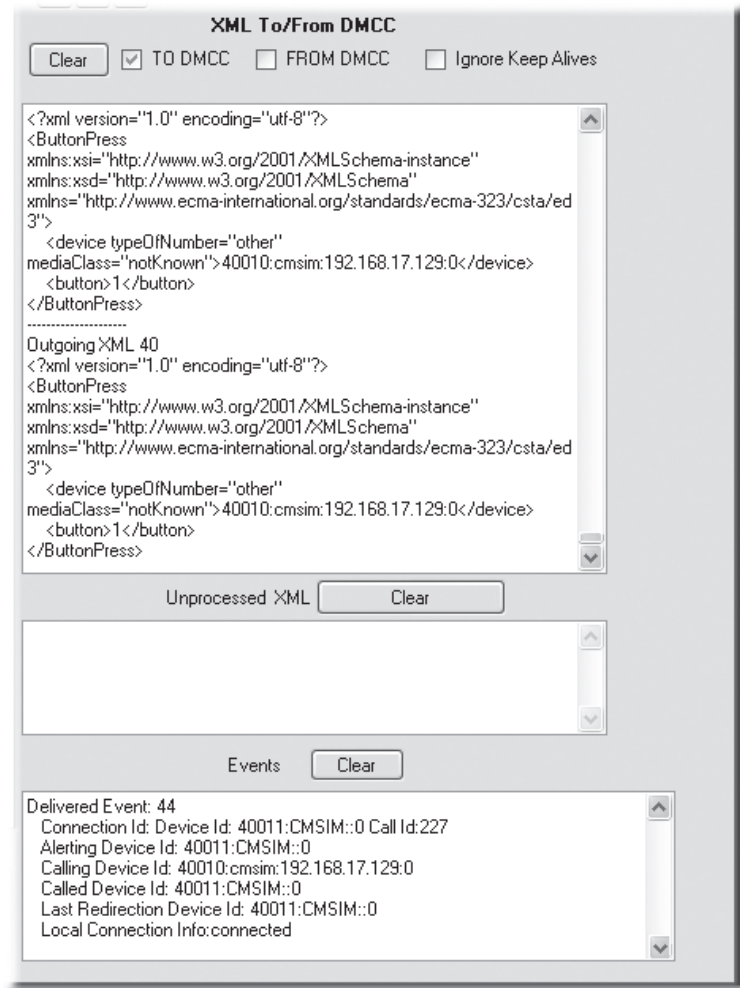

Figure 3-4: XML Messages and Events Region

Region 4: Exceptions and Errors

The Exceptions and Errors region of the DMCC Dashboard incorporates a scrollable text box that displays details of exceptions and errors returned by the DMCC service.

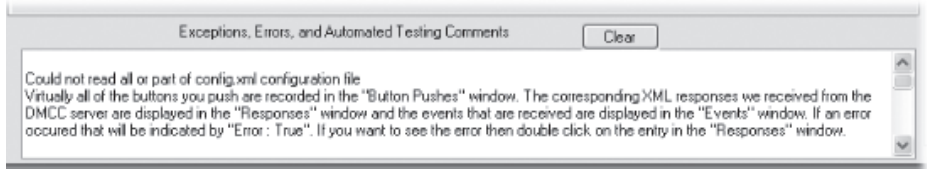

Figure 3-5: The Exceptions and Errors Region

Usability Features and Help

The DMCC Dashboard provides various features and includes context-sensitive tooltips and different levels of on-line help to make using the tool as easy and as intuitive as possible. A comprehensive user guide is also provided with the DMCC Dashboard download.

Intelligent Control Activation

Controls only become active and available for use when the DMCC Dashboard is in the appropriate state and all prerequisite actions have been performed. Inactive controls are grayed out. For example, the **Get Device ID** button on the **Main** tab does not become active until a session is started, and the DMCC softphone controls only become active once the DMCC Dashboard has registered itself as one or more DMCC devices and one of the associated Device IDs is selected.

Identification of Fields Related to Actions

When users place their mouse over an active button, the fields related to the button action are highlighted. The highlighted fields are not necessarily mandatory, but the values in them will have an effect on the action and so should be considered by the user before pressing the button.

Figure 3-6 shows **Start Application Session**-related fields being highlighted when the user's mouse hovers over the **Start Appl. Session** button.

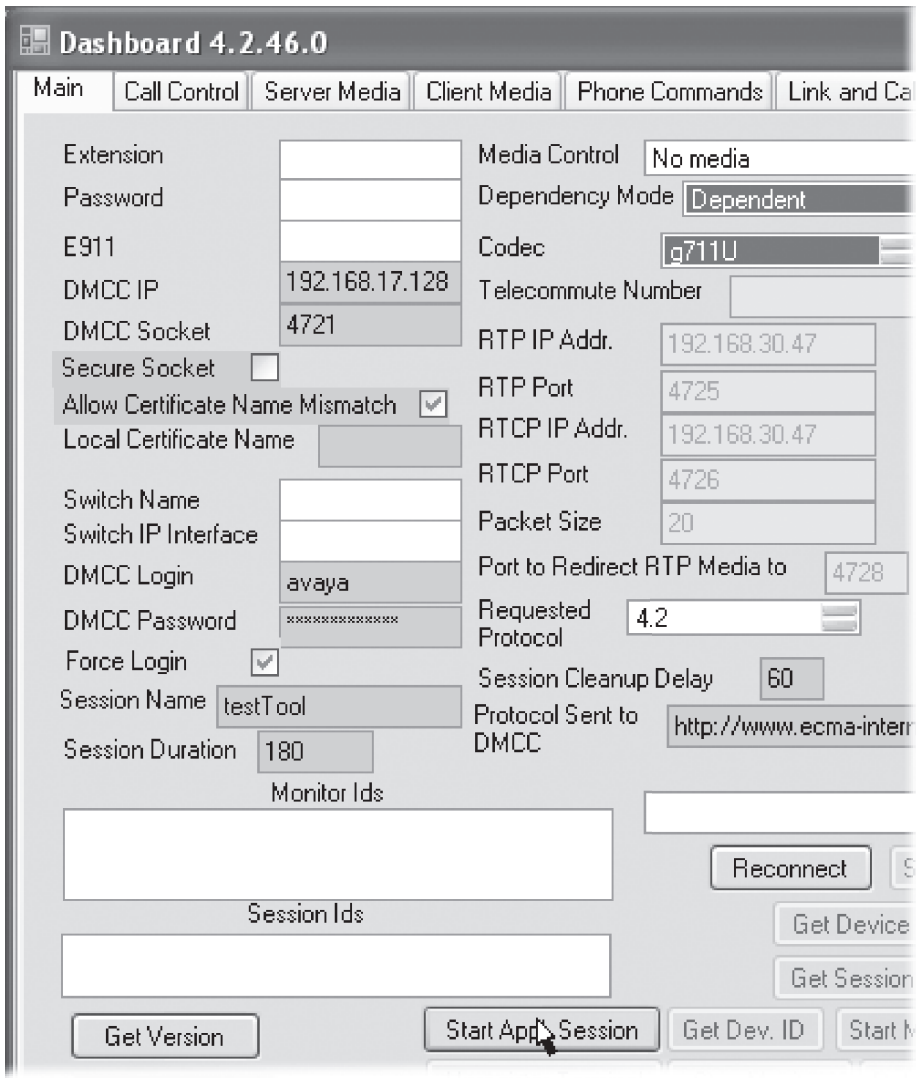

Figure 3-6: The **Start Application Session** Fields

Tooltips

Most of the controls on the DMCC Dashboard have tooltips associated with them. A brief description is displayed when users place their mouse over a control. **Figure 3-7** shows the tooltip displayed for the **Reconnect** button on the **Main** tab.

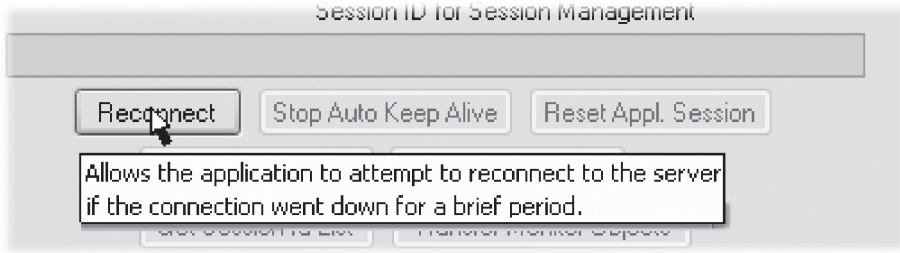

Figure 3-7: Tooltip for the **Reconnect** Button

On-line Help

Many of the controls also have more detailed on-line help associated with them. To access the help, click on the **?** icon in the top right-hand corner of the DMCC Dashboard window and then click on the control. If help is available, it is displayed in a pop-up dialog. **Figure 3-8** shows the on-line help dialog for the **Reconnect** button.

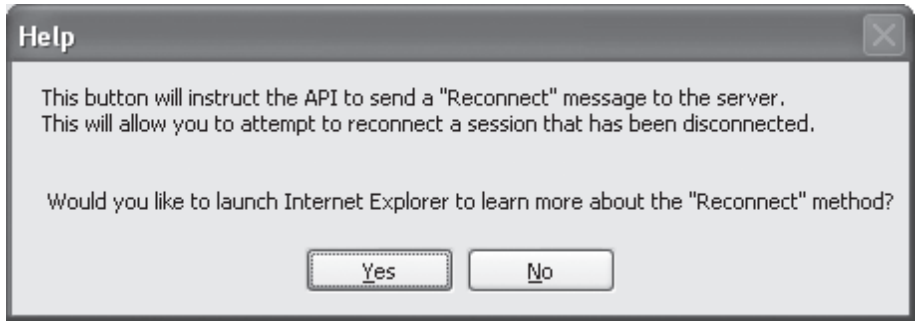

Figure 3-8: On-line help for the **Reconnect** Button

Where relevant, users can also view web-based API documentation for the method called by the control. If available, a prompt is included at the bottom of the on-line help dialog: click **Yes** to open the API documentation in your web browser. **Figure 3-9** shows the API documentation for the Reconnect method.

Note: The API documentation is hosted on the DevConnect web server at the location shown in the **Path to online documentation** field on the **Main** tab.

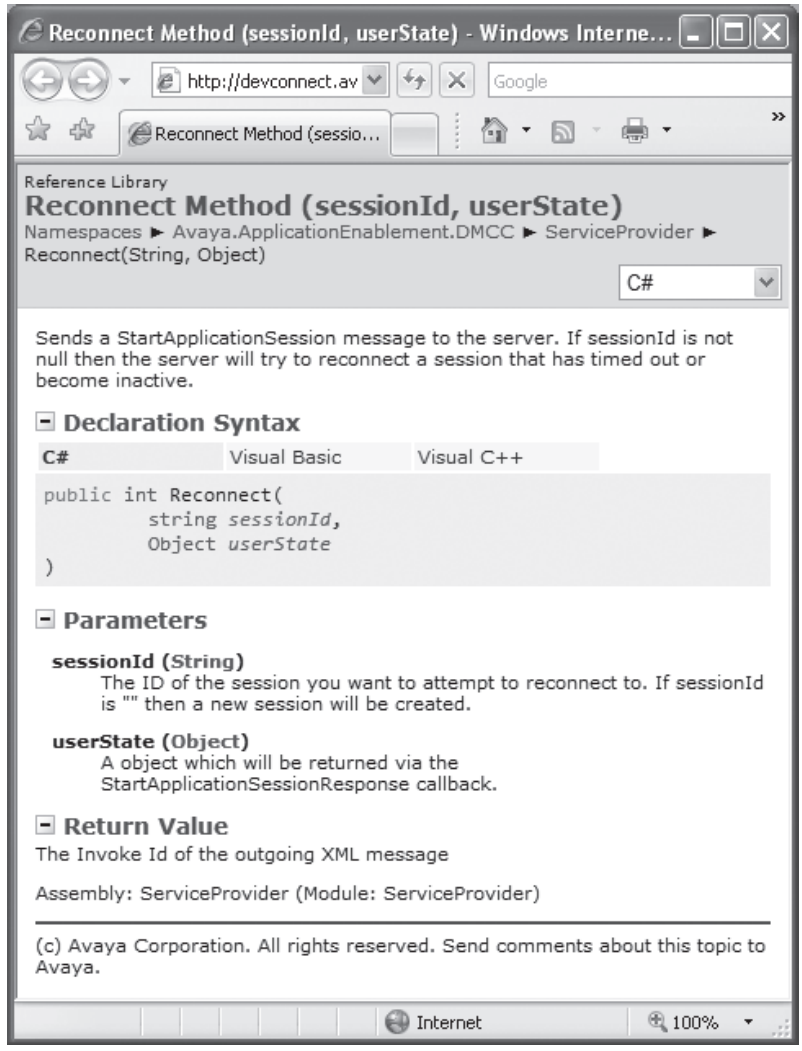

Figure 3-9: Web-based Help for the Reconnect Method

Round-up

In this, the final chapter of Part I of the book, you learned about the layout of the DMCC Dashboard user interface and other features designed to make it easy to use. You also learned about the various levels of on-line help available and how to access it.

You are now ready to go on to Part II where you will explore each of the DMCC Service Control tabs in detail and learn how to use the DMCC Dashboard to exercise the capabilities of the DMCC service.

Part 2:
Using the DMCC Dashboard

What's in Part II?

In Part II, you will learn how to use the DMCC Dashboard to exercise the capabilities of the DMCC service, including physical device control, client- and server-side media control and basic third-party-call control. You will also discover how the DMCC Dashboard can be used to prototype and test applications under development. The chapters in this part are organized based on the tabs of the DMCC Dashboard user interface.

By the end of Part II, you'll have the knowledge you need to begin using the DMCC Dashboard.

36

Chapter 4

Sessions, Monitors and Device Registration

In this chapter:

- About application sessions, event monitors and DMCC device registration
- Exploring the **Main** tab
- Quick start guide to using the DMCC Dashboard
- Starting and managing application sessions
- Getting Device IDs to identify and control extensions
- Starting and managing event monitors
- Registering DMCC devices for first-party device and media control
- Converting E.164 format telephone numbers
- Performing basic first-party call control using the DMCC Dashboard IP softphone
- Sending raw XML messages directly to the AE Services DMCC service
- Configuring the DMCC Dashboard

As you saw in *Chapter 3*, the DMCC Service Controls region of the DMCC Dashboard comprises a series of tabbed panels. The tabs group sets of related options that allow users to access and exercise AE Services DMCC service functionality. Most of the tabs contain options that are specific to a particular type of device, media or call control. For example, the **Call Control** tab contains the options used for third-party call control. The **Main** tab differs from the other tabs in that it contains more generic options used for a number of different purposes. This chapter is about the options available on the **Main** tab.

Among the most important options on the **Main** tab are those used to perform common tasks that are a prerequisite to exercising device, media and call control capabilities at the other tabs. These common tasks involve some or all of the following:

- Starting application sessions
- Getting the first-party Device IDs of the extensions you want to monitor or control
- Starting event monitors
- Registering DMCC devices at the extensions you want to control

The section *Quick Start Guide* in this chapter provides a brief overview of these common tasks and tells you which are required and which are optional for each type of device, media and call control. The individual options used to perform each of the common tasks are described in detail in the sections following the *Quick Start Guide*.

The **Main** tab also contains options that allow you to manage sessions, configure how the DMCC Dashboard works, send raw XML messages to the DMCC service, convert telephone numbers formats and so forth.

This chapter incorporates detailed descriptions of all the options on the **Main** tab, plus the DMCC Dashboard IP softphone. This chapter also provides information and advice on how the underlying functionality can be leveraged in DMCC client applications you develop.

For the examples in this chapter, and in most of the chapters that follow, the required Avaya Communication Manager and Avaya AE Services installation is provided by the Avaya IP Communications Development Environment (Avaya IPCoDE), installed on the same desktop machine as the DMCC Dashboard. See *Chapter 2* for more information about Avaya IPCoDE and alternative options for accessing an Avaya Communication Manager and Avaya AE Services installation. The examples use the virtual DCP telephones provided by Avaya IPCoDE (at extensions 40010, 40011 and 40012) and an Avaya IP Softphone logged into extension 32129.

The Main Tab

Figure 4-1 shows the **Main** tab when the DMCC Dashboard is first started.

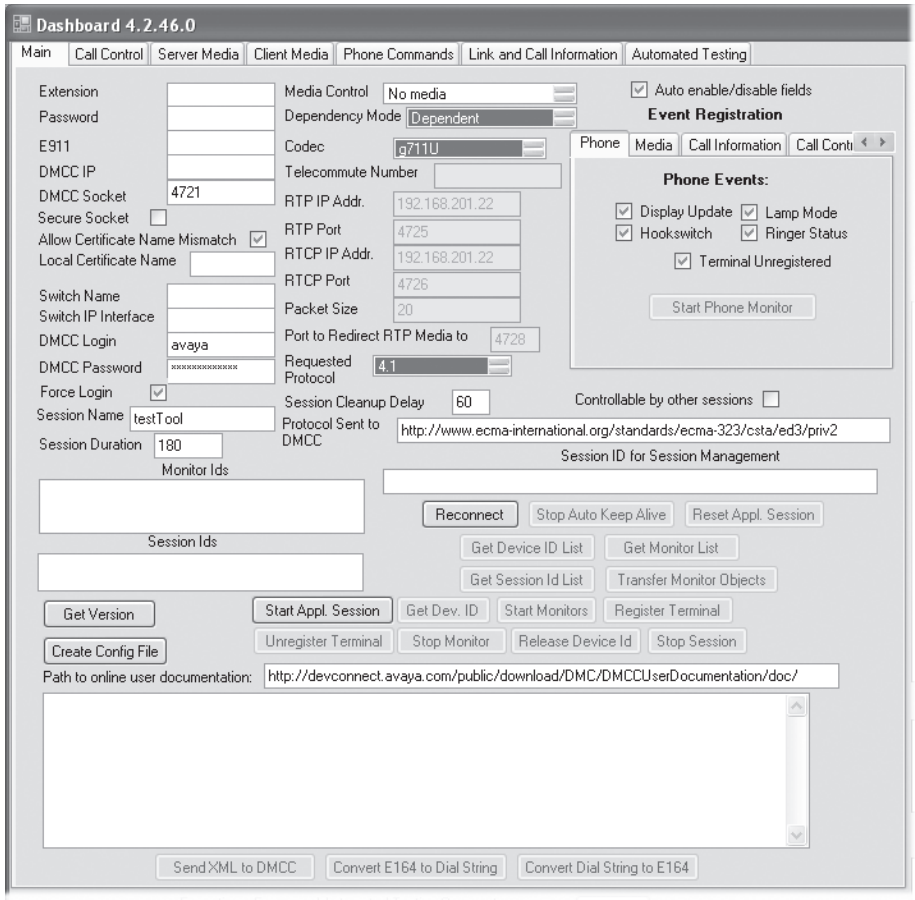

Figure 4-1: The DMCC Dashboard **Main** Tab

Many of the fields are populated with default values. See *Configuring the DMCC Dashboard* at the end of this chapter for instructions on how you can change the default values displayed when you first start the DMCC Dashboard.

At first sight the **Main** tab looks very busy, but by the time you reach the end of this chapter you will know your way around it and be familiar with all of the options it contains.

Quick Start Guide

As mentioned in the introduction to this chapter, some of the most important options on the **Main** tab are used to perform common tasks as a prerequisite to using device, media and call control options available at the other tabs.

All DMCC client applications you create will have to perform most, if not all, of these tasks.

For convenience, the option buttons used to perform the prerequisite tasks are aligned in a row in the order in which they must be used.

Figure 4-2: Quick Start Options

The option buttons only become available when the previous prerequisite tasks have been completed. The options are:

- **Start Appl. Session:** Used to start an application session that allows the DMCC Dashboard to communicate with the AE Services DMCC Service. Starting an application session is mandatory for all types of device, media and call control.
- **Get Dev. ID:** Used to get the first-party Device ID of an extension you want to monitor or control. Getting a Device ID is mandatory if you want to start Phone, Media and Call Control event monitors at an extension, or want to control the phone set or media at an extension.
- **Start Monitors:** Used to start monitors that allow the DMCC Dashboard to listen for Phone, Media and Call Control events at an extension, and to listen for Call Information and Session Management events for the whole application session. Starting monitors is not mandatory for any type of device, media or call control. However, being able to see the events generated helps you understand the functionality exercised on the DMCC Dashboard. In addition, any DMCC client applications you create will need to listen for appropriate events.
- **Register Terminal:** Used to register the DMCC Dashboard as a DMCC device at an extension. Registering a DMCC device at an extension is mandatory if you want to control its media or perform first-party device control.

The chapters that follow describe a particular type of device, media or call control. Each chapter includes a *Preliminary Steps* section that reminds you of the prerequisite tasks you need to perform at the **Main** tab.

At the end of a session, use the buttons on the row below the prerequisite task options to clean up by unregistering DMCC devices, stopping monitors, releasing first-party Device IDs and stopping the session.

Each of the prerequisite task options is described in detail in the sections that follow.

Starting and Stopping Application Sessions

Before you can exercise DMCC service capabilities you must start an application session with the AE Services server. Application sessions are established between client applications (in this case, the DMCC Dashboard) and the AE Services server to allow application messages to be exchanged between them.

Starting an application session involves connecting the client application to the AE Services server, configuring various aspects of the session's behavior and obtaining a Session ID that can subsequently be used to identify and manage the session.

This section describes how to start sessions, keep sessions alive, recover from session failures and stop sessions.

Starting Application Sessions

DMCC Dashboard application sessions are started by supplying the required field values and clicking **Start Appl. Session**. You'll remember from *Chapter 3* that when you place your mouse over a control, the associated fields are highlighted. **Figure 4-3** shows the fields associated with starting an application session.

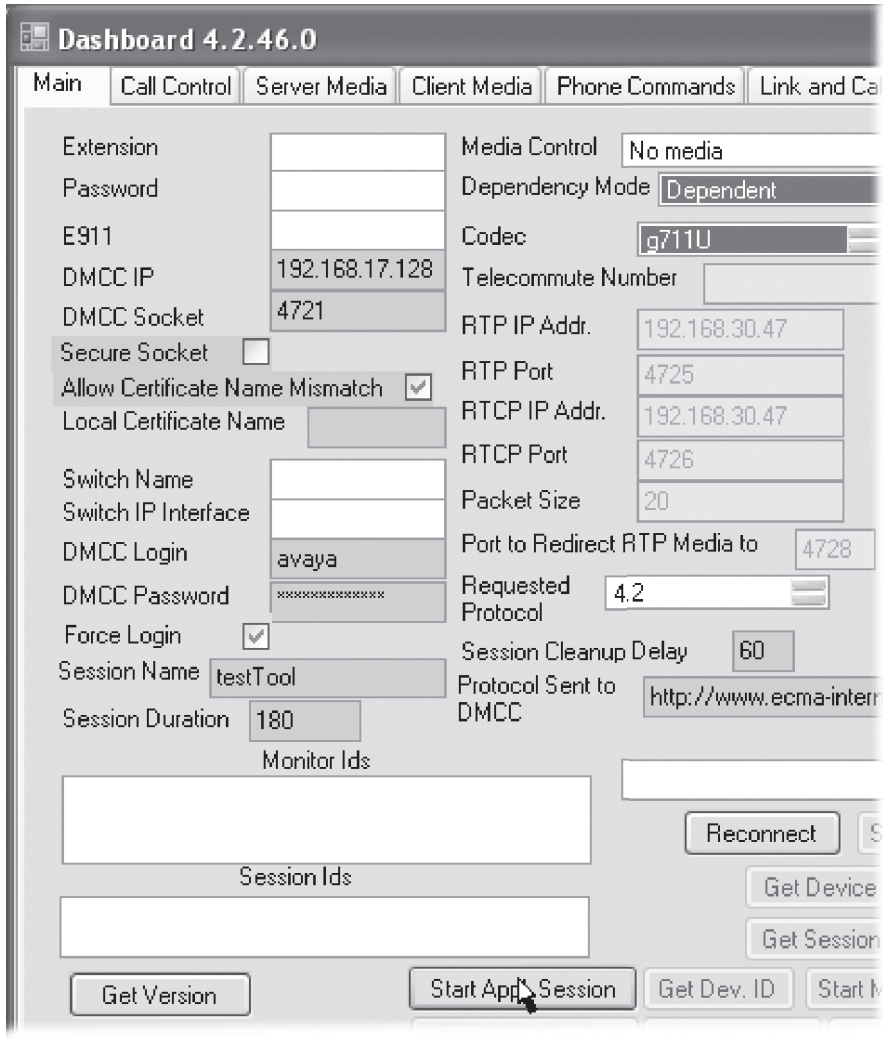

Figure 4-3: Starting an Application Session

The values in the highlighted fields are used when an application session is started:

DMCC IP and **DMCC Socket:** The IP address or DNS name of the AE Services server and the port used to access the DMCC service. For Avaya IPCoDE the DMCC IP address is "192.168.17.128". The DMCC service port is either "4721" or "4722" depending on whether the DMCC Dashboard is to be connected using an unsecure or a secure socket.

Secure Socket: Check if using a secure socket. To use a secure socket, a security certificate must be installed. The Avaya security certificate (`avaya.crt`) is provided with the DMCC .NET SDK.

Allow Certificate Name Mismatch: Check if a secure socket is to be used even if the specified security certificate does not exactly match the required certificate.

Local Certificate Name: The name of the security certificate to be used, if using a secure socket. Leave blank if using the `avaya.crt` certificate provided with the DMCC .NET SDK.

DMCC Login and DMCC Password: The username and password used to log into the DMCC service to which the DMCC Dashboard application session is to be connected.

Session Name: A user-friendly name given to the application session to help identify it.

Session Duration: The period, in seconds, after which the DMCC service will disconnect the application session if it does not receive a "keep alive" heartbeat from the DMCC Dashboard.

Session Cleanup Delay: The period, in seconds, after which an application session will be terminated once it has been disconnected from the DMCC service. During this period, you can attempt to reconnect the session on a different port by clicking **Reconnect.**

Protocol Sent to DMCC: The namespace string that represents the version of the DMCC XML protocol to be used. The value in this field is updated automatically, based on the version of AE Services you are using, and does not normally need to be changed. Specify the AE Services version by selecting and highlighting it in the **Requested Protocol** field.

In practice, if you want to use the security certificate provided with the DMCC .NET SDK, you only need to supply the IP address of the AE Services server; all the other fields default to acceptable values.

By default, AE Services does not allow communication via an unsecure socket. If you wish to use an unsecure socket, you must configure your AE Services installation accordingly.

When attempting to start an application session, the DMCC Dashboard sends a `StartApplicationSession` XML message to the DMCC service. If all goes well, the DMCC service responds with a `StartApplicationSessionPosResponse` XML message. These messages are displayed in the **XML To/From DMCC** scrollable text box in the XML Messages and Events region on the right-hand side of the DMCC Dashboard user interface, as shown in **Figure 4-4**.

The `StartApplicationSession` method automatically opens the socket to the AE Services server. You do not need to be concerned with any socket management.

The full text of the XML messages is shown on the next page. You can clearly see the session configuration parameters passed in the outgoing request message and the unique Session ID returned by the DMCC service on the incoming response message.

In addition to the full XML messages displayed in the **XML To/From DMCC** field, the DMCC Dashboard extracts and displays summary details in the **Events** text box.

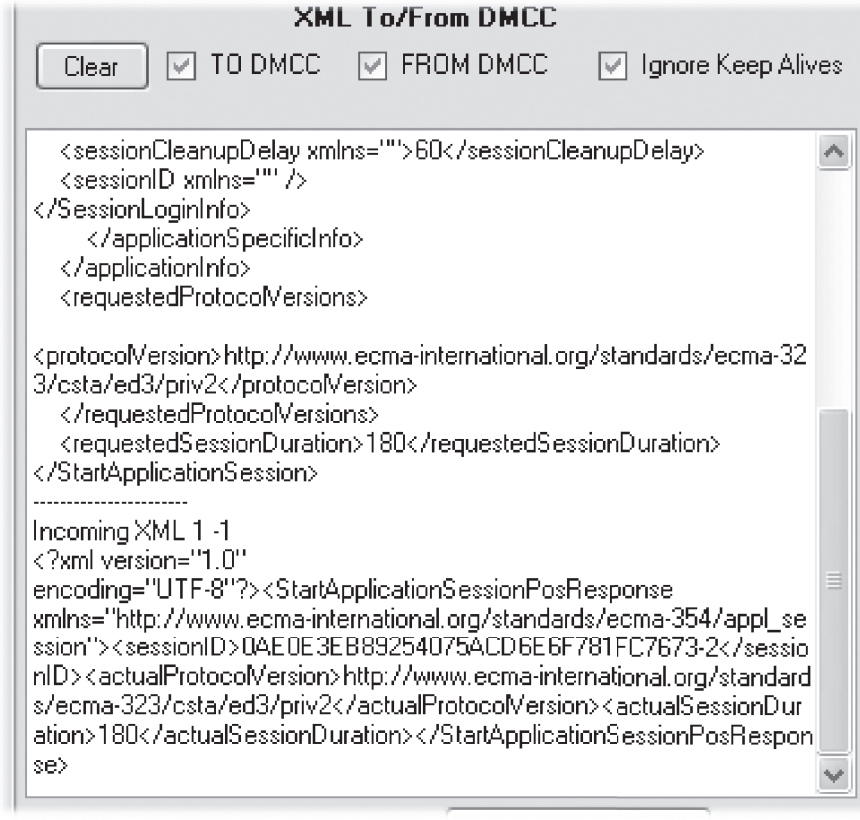

Figure 4-4: Start Application Session XML Messages

Start Application Session: Outgoing and Incoming XML Messages

```
Outgoing XML 1
<?xml version="1.0" encoding="utf-8"?>
<StartApplicationSession  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://
www.ecma-international.org/standards/ecma-354/appl_session">
    <applicationInfo>
        <applicationID>myApplicationSession</applicationID>
        <applicationSpecificInfo>
            <SessionLoginInfo xmlns="http://www.avaya.com/csta">
                <userName xmlns="">avaya</userName>
                <password xmlns="">avayapassword</password>
                <sessionCleanupDelay xmlns="">
                     60
                </sessionCleanupDelay>
                <sessionID xmlns="" />
             </SessionLoginInfo>
        </applicationSpecificInfo>
    </applicationInfo>
    <requestedProtocolVersions>
        <protocolVersion>
             http://www.ecma-international.org/standards/ecma-
             323/csta/ed3/priv2
        </protocolVersion>
    </requestedProtocolVersions>
    <requestedSessionDuration>180</requestedSessionDuration>
</StartApplicationSession>
-----------------------
Incoming XML 1 -1
<?xml version="1.0" encoding="UTF-
8"?><StartApplicationSessionPosResponse xmlns="http://www.ecma-
international.org/standards/ecma-354/appl_session"><sessionID>275FDAE
2FE8E051A71AD3C79D34B7BA6-0</sessionID><actualProtocolVersion>http://
www.ecma-international.org/standards/ecma-323/csta/ed3/priv2</actual
ProtocolVersion><actualSessionDuration>180</actualSessionDuration></
StartApplicationSessionPosResponse>
```

The Session ID is displayed in the **Session IDs** list box on the **Main** tab.

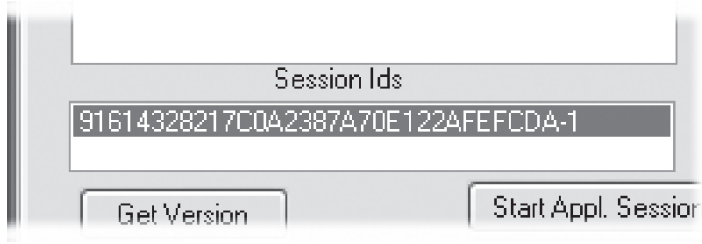

Figure 4-5: Session IDs List Box

By default, all XML messages exchanged between the DMCC Dashboard and the DMCC service, except "keep alive" messages, are displayed in the **XML To/From DMCC** text box. Incoming or outgoing XML messages can be filtered out, or "keep alive" messages included, by deselecting the appropriate check boxes above the text box.

Being able to see the generated XML messages is particularly valuable if you want to use the DMCC XML protocol and SDK to create DMCC client applications. You can see the outgoing XML messages that your applications must generate to submit requests to the server.

Keeping Sessions Alive

When you start a new application session, the DMCC client application automatically begins sending periodic `ResetApplicationSessionTimer` messages to the DMCC service to keep the session alive. If the DMCC service does not receive a "keep alive" message within the specified Session Duration period, the session is disconnected from the server and, unless it is reconnected within the specified Session Cleanup Delay period, the session is terminated.

Normally, sessions are disconnected and then terminated when the client application is shut down or becomes unavailable for some other reason. However, you can emulate this behavior from the DMCC Dashboard. To stop sending "keep alive" messages, click **Stop Auto Keep Alive**. Note that this button toggles the "keep alive" behavior, so the label changes to **Start Auto Keep Alive** after it has been clicked.

Now that "keep alive" messages are not being sent automatically, you can click to send them manually or allow the session to become disconnected. While in the disconnected state, and before the session is terminated, you can click **Reset Appl. Session** to attempt to reconnect the session on a different port, or create a new session and transfer devices and monitors to it—see *Starting Multiple Sessions* below.

To restart automatic "keep alive" messages, click **Start Auto Keep Alive**.

You can use these features to help build automatic session recovery into your applications.

If your session application stops, use the **Start Appl. Session** option to start another session with a new Session ID. You will also need to get Device IDs, restart monitors and reregister DMCC Devices that you were using.

Starting Multiple Sessions

You can start and manage multiple application sessions from the DMCC Dashboard by simply entering the required field values and clicking **Start Appl. Session,** as required. The IDs of all active sessions are listed in the **Session IDs** list box. As you will see, some session management options require you to enter this ID in the **Session ID for Session Management** field.

Multiple application sessions can be used by client applications to recover from session failure. If a session becomes disconnected, the client application can start a new session and transfer Device IDs, event monitors and DMCC device registrations to the new session. The new session will receive information enabling it to process events without starting new monitors. See *Transferring Monitors* later in this chapter for information on how to demonstrate session recovery using the DMCC Dashboard.

Getting Session IDs

To retrieve a list of sessions currently running, click **Get Session ID List**. A list of active Session IDs is displayed in the **Events** text box.

Session IDs are long, complex strings and it is easy to make a mistake typing them into the **Session ID for Session Management** field. A quicker and more reliable method is to get a list of active Session IDs, then copy and paste the one you want from the **Events** text box.

Stopping Sessions

To end a session, click to highlight it in the **Session IDs** list box and click **Stop Session**.

Before stopping a session you should clean up by unregistering all DMCC devices, stopping all monitors and releasing all first-party Device IDs.

Getting First-party Device IDs

Once an application session has been started, you need to get the first-party Device IDs associated with the extensions you want to monitor and control. You must get the first-party Device IDs of extensions at which you want to start Phone, Media or Call Control event monitors. You must also get the first-party Device IDs of extensions at which you want to control the media (server- or client-side), or at which you want to use first-party device control.

First-party Device IDs are retrieved using the **Get Dev. ID** button. **Figure 4-6** shows the fields that need to be considered when getting the Device ID for extension 40010.

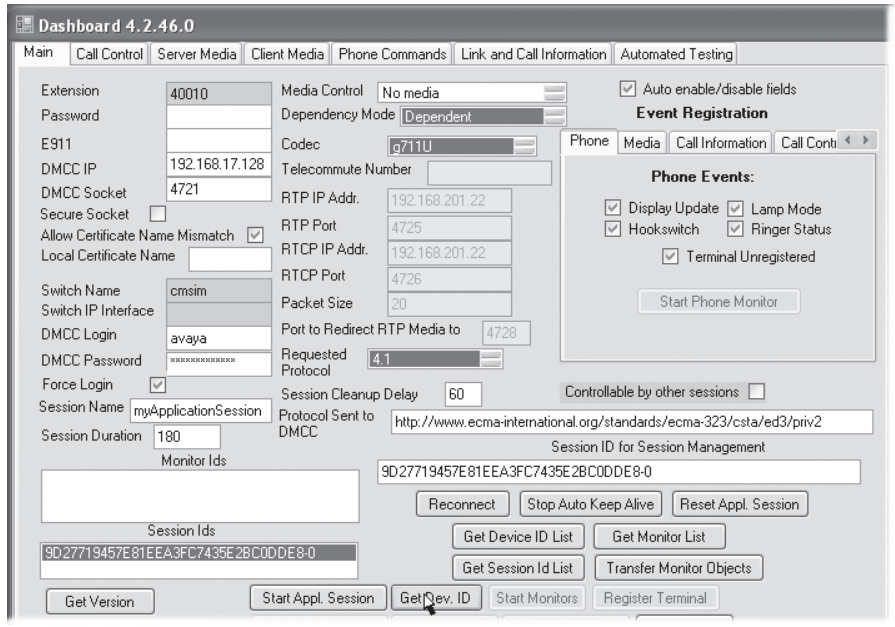

Figure 4-6: Getting Device IDs

The values in the highlighted fields are used when getting first-party Device IDs:

Extension: The number of the extension for which you want to retrieve the Device ID.

Switch Name or Switch IP Interface: The name or IP number of the AE Services switch connection. For Avaya IPCoDE, the Switch Name is "cmsim" and Switch IP Interface "192.168.17.129": enter either value, but not both.

Controllable by other sessions: Check if you want to allow the extension to be controllable by other application sessions. If checked, other sessions will be able to get first-party Device IDs that have already been retrieved in the current session. This opens up the possibility of multiple sessions being able to listen for events at extensions, and also of being able to share control of the media and phone sets at extensions. If not checked, attempts by other sessions to get first-party Device IDs will fail if the Device IDs have already been retrieved in the current session.

When the **Get Dev. ID** button is clicked, a `GetDeviceId` message is sent to the server which responds with a `GetDeviceIdResponse` message containing the Device ID. The Device ID is displayed in the **Device IDs** list box to the right of the DMCC Dashboard IP softphone, as shown in **Figure 4-7.**

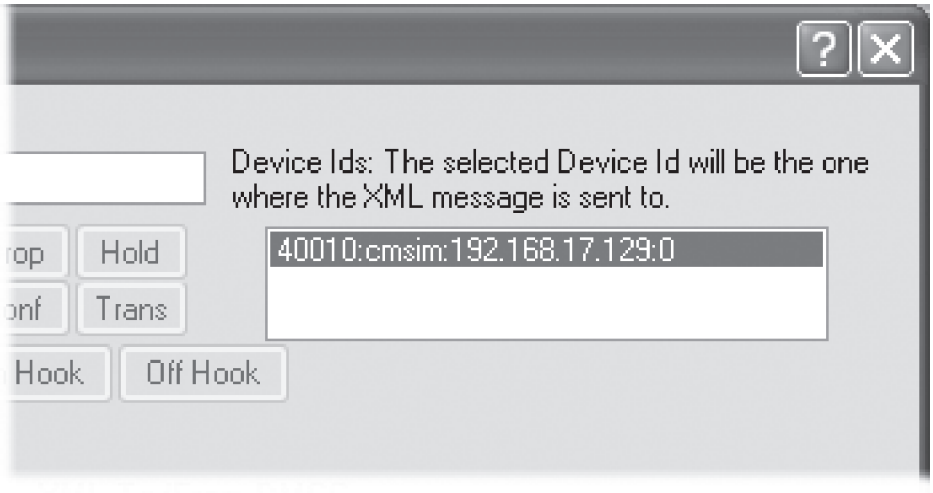

Figure 4-7: Device IDs List Box

As you can see, in this case, the Device ID has the format:

[Extension no.]:[Communication Manager name]:[Communication Manager IP]:0

The Device ID retrieved for each extension you want to control is displayed in the **Device IDs** list box. The highlighted Device ID is the one that XML messages will be sent to when you use the DMCC Dashboard to exercise various capabilities of the DMCC service.

A different type of Device ID is usually used for third-party call control, namely third-party Device IDs. However, any third-party call control applications you create will almost certainly need to listen for Call Control and other events at the extensions it is controlling. Therefore, you will still have to get first-party Device IDs for those extensions. See *Chapter 5: Third-party Call Control* for information about third-party Device IDs and how they differ from first-party Device IDs.

Listing Device IDs

You can obtain information about the first-party Device IDs that have been retrieved for a particular application session by entering the Session ID in the **Session ID for Session Management** field and clicking **Get Device ID List**. However, this option will only work correctly if you have started the Session Management monitor to listen for Device ID List events—see the next section on *Setting and Managing Monitors*.

If the Session Management monitor has been started, details of the generated `GetDeviceIdList` event are displayed in the **Events** text box at the bottom-right of the DMCC Dashboard. The event includes a list of Device IDs for the session.

Releasing Device IDs

When your application has finished controlling extensions, or is about to close, it should release the extensions' first-party Device IDs. You can demonstrate this on the DMCC Dashboard by selecting a Device ID in the **Device IDs** list box and clicking **Release Device ID**. A `ReleaseDeviceId` message is sent to the server.

It is essential that client applications *clean up* when they have finished controlling extensions and before the application session is ended. Before releasing a Device ID, the client application must unregister DMCC devices and stop monitors. See *Stopping Monitors and Unregistering DMCC Devices* later in this chapter for information about how to use the DMCC Dashboard to demonstrate clean-up functionality.

Starting and Managing Event Monitors

Once you have retrieved the first-party Device IDs you want to control, you can start monitors that allow the DMCC Dashboard to listen for Phone, Media and Call Control events at the corresponding extensions. You can also start monitors that allow the DMCC Dashboard to listen for all Call Information and Session Management events generated during the application session.

As you would expect, when a DMCC client receives notification of an event, the corresponding callback method is called automatically. For example, if using the DMCC Java API, when a call is established at an extension, the application receives an `Established` event and calls using the `established(EstablishedEvent)` method. The callback method may start call recording at the extension; or perform some other functionality determined by the application. The callback methods implemented by the DMCC Dashboard simply extract information from the event and display it in the **Events** text box.

Monitoring Events

To listen for particular events, a client application has to start the appropriate event monitors. Some monitors allow the application to listen for events at particular extensions; other event monitors allow the application to listen for session-level events.

To monitor Phone, Media or Call Control events at a particular extension, you must select the extension's Device ID in the **Device IDs** list box, select the events you want to listen for and start the appropriate monitors.
To monitor Call Information and Session Management events generated at session level, you just select the events you want to listen for and start the appropriate monitors.

Events are selected using the **Event Registration** tabs on the **Main** tab. Each **Event Registration** tab lists events of a particular type:

- **Phone**
- **Media**
- **Call Information**
- **Call Control**
- **Session Mgmt.**

By default, all the events are selected. If you don't want to monitor a particular event, simply deselect it. You can start listening for events of a particular type by clicking **Start ... Monitor** on the appropriate tab, or start listening for events of all types by clicking **Start Monitors** next to the **Get Dev. ID** button.

Figure 4-8 shows all the **Event Registration** tabs and **Table 4-1** describes the circumstances under which each of the events is generated.

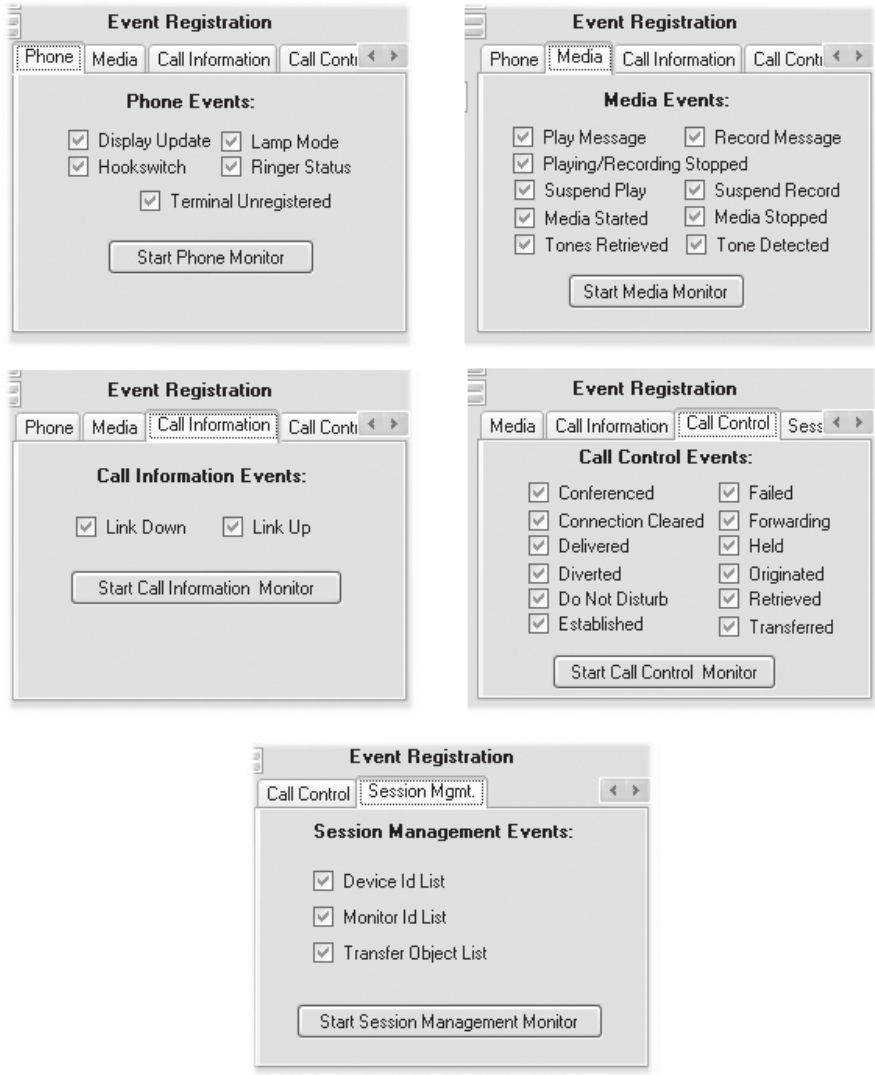

Figure 4-8: Event Registration Tabs

Table 4-1
Monitored Events

Phone Events	
Display Update	The display at the monitored extension has changed.
Hookswitch	Communication Manager has changed the monitored extension's hookswitch status because other parties have dropped out of a call.
Lamp Mode	The state of one of the lamps at the monitored extension has changed. For example, the lamp has started flashing, fluttering, shining steadily or has gone off.
Ringer Status	The ringer at the monitored extension has started or stopped ringing.
Terminal Unregistered	The DMCC Dashboard is no longer registered as a DMCC device at the monitored extension.
Media Events	
Play Message	A message is being played on the monitored extension's voice unit.
Playing/Recording Stopped	The playing or recording of a message at the monitored extension has stopped for some reason.
Suspend Play	The message being played at the monitored extension has been suspended.
Media Started	An RTP media session has been established at the monitored extension.
Tones Retrieved	The monitored extension has retrieved a DTMF digit from the buffer
Record Message	A message is being recorded at the monitored extension.
Suspend Recording	A message has been suspended during recording at the monitored extension.
Media Stopped	An RTP media session at the monitored extension has been disconnected.
Tone Detected	The monitored extension has received a DTMF digit.
Call Information Events	
Link Down	The communications link between Communication Manager and AE services that allows detailed call information to be retrieved has stopped operating.
Link Up	The call information link has become operational.

Table 4-1
Monitored Events Cont.

All Call Control Events	
Conferenced	The monitored extension has added itself to a conference call, or has added another extension in an existing call to a conference call.
Connection Cleared	The monitored device has disconnected or dropped from a call.
Delivered	A call is being presented to the monitored device, either in the Ringing or Entering Distribution mode of the alerting state.
Diverted	A call has been diverted from the monitored extension and is no longer at that extension.
Do Not Disturb	The "do not disturb" feature has been changed at the monitored extension.
Established	A call has been answered at, or connected to, the monitored extension.
Failed	An outgoing dialog (typically from a client application `MakeCall` request) has failed to be connected at the monitored extension, for some reason.
Forwarding	The forwarding feature has been changed at the monitored extension.
Held	A call involving the monitored extension has been put on hold.
Originated	An attempt is being made to make a call from the monitored extension.
Retrieved	A previously held call at the monitored extension has been taken off hold.
Transferred	The monitored extension has transferred a call to another extension and has been dropped from the call.
Session Management	
Device ID List event	A `GetDeviceIdList` request has been made for a session. The returns a list of Device IDs.
Monitor ID List event	A `GetMonitorList` request has been made for a session. The returns a list of monitors that have been set for the service.

Once you start the event monitors, the DMCC Dashboard is automatically notified whenever one of the monitored events occurs and displays its details in the **Events** text box. **Figure 4-9** shows the event details displayed when you start the Session Management monitor and click **Get Device ID List.**

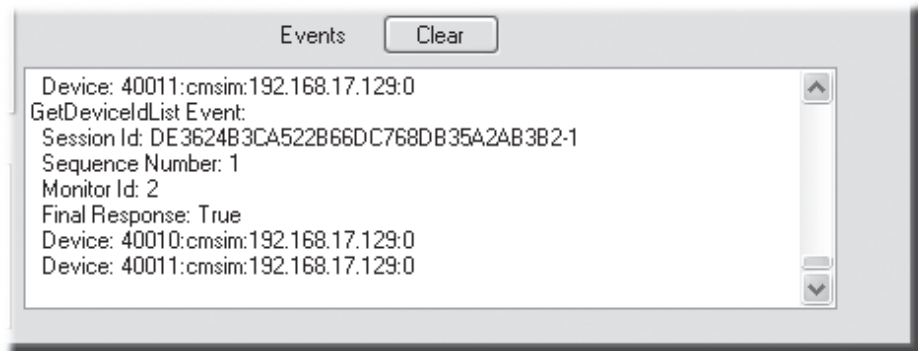

Figure 4-9: **GetDeviceIdList** Event

Details of all active event monitors are displayed in the **Monitor IDs** list box on the **Main** tab.

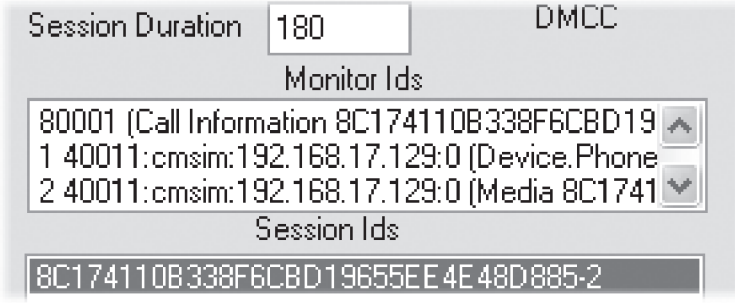

Figure 4-10: Monitor IDs List Box

Managing Monitors

Once you have started one or more monitors in an application session, you can use options on the **Main** tab to:

- Retrieve a list of active monitors
- Transfer the monitors to a different application session
- Stop the monitors

Viewing Active Monitors

To retrieve a list of active monitors for an application session, ensure the Monitor ID List event monitor has been started, enter the Session ID in the **Session ID for Session Management** field, and click **Get Monitor List**. A `GetMonitorList` event is generated and details of the monitors displayed in the **Events** text box.

Transferring Monitors

If you have multiple active sessions, you can transfer Device IDs, their monitors and DMCC device registrations from one session to another. To transfer monitors, enter the Session IDs of the "from" and "to" sessions in the **Session ID for Session Management** field (separated by a space) and click **Transfer Monitor Objects**.

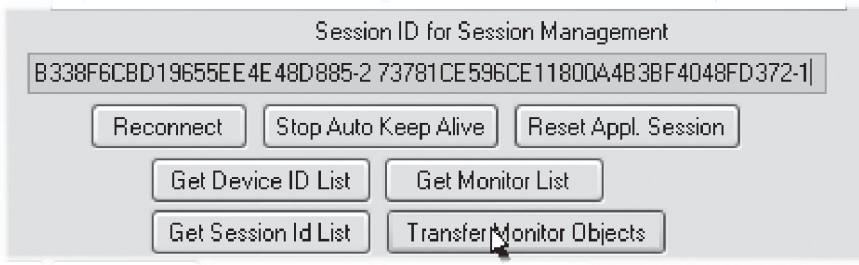

Figure 4-11: Transferring Monitors

A `TransferMonitorObjects` message is sent to the server. The monitors are transferred to the "to" session and the "from" session is automatically ended.

You can use this option to emulate how your application might recover from session failure by starting a new session and transferring Device IDs, event monitors and DMCC device registrations to it.

Stopping Monitors

To stop monitors of a particular type (Phone, Media, Call Information, etc.) at a particular extension, click to highlight the appropriate entry in the **Monitor IDs** list box and click **Stop Monitor**, as shown in **Figure 4-12**. A `StopMonitor` message is sent to the server.

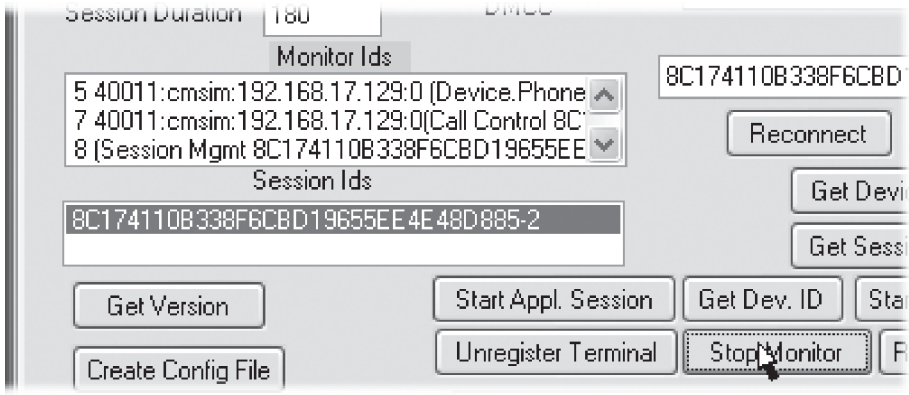

Figure 4-12: Stopping Monitors

Registering DMCC Devices

To be able to perform first-party device control at an extension, or to manage the media streams at an extension, a client application (in this case, the DMCC Dashboard) must first register itself as a DMCC device at the extension. See *Chapter 1: Device Registration* for more information about DMCC device registration, including supported extension types and multiple device registration.

A DMCC client application **does not** need to register DMCC devices at extensions to be able to perform third-party call control at those extensions.

Registering a DMCC Device at an Extension

Once you have got the first-party Device ID of an extension (see *Getting Device IDs* above) you can register a DMCC device at that extension by selecting its Device ID in the **Device IDs** list box and clicking **Register Terminal**.

The field values that need to be considered when you register a DMCC device vary depending on the type of control you want to have, as determined by the selected Media Control mode. **Figure 4-13** shows the location of the **Register Terminal** button on the **Main** tab and the fields you need to consider if you only wish to control the physical aspects of the phone set at the extension but not its media, i.e. with the Media Control mode set to “No Media”.

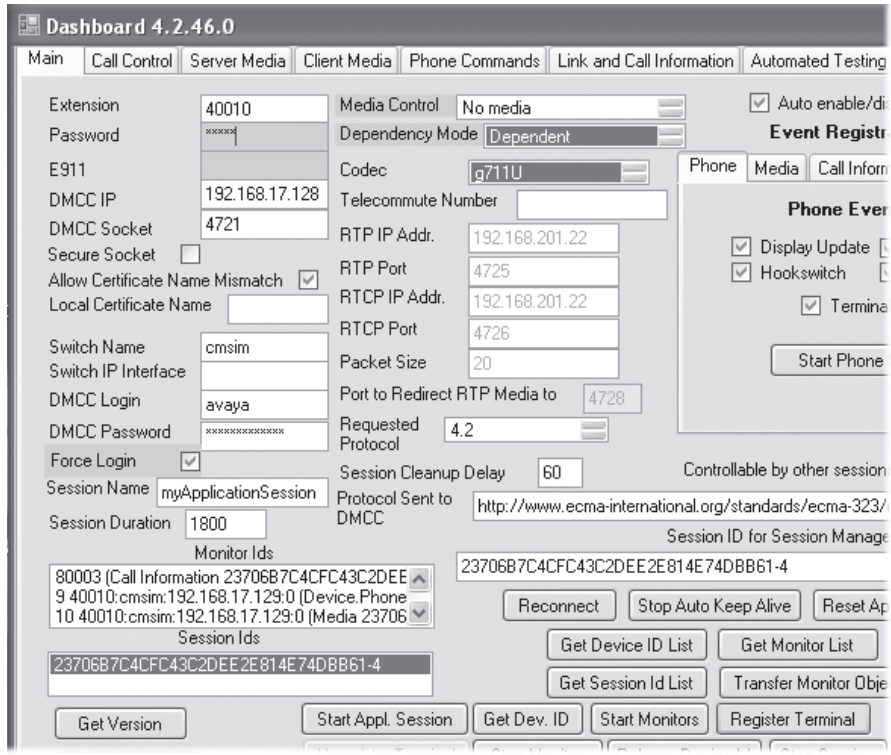

Figure 4-13: Registering DMCC Devices

The values in the highlighted fields are used when registering DMCC devices in No Media or Server Media modes.

Password: The password used to log into the extension. For the Avaya IPCoDE DCP telephones and Avaya H.323 IP softphones, the password is the same as the extension number.

E911: The phone number associated with this extension, to be used when you dial 911 to make an emergency call. Typically, this value is left blank so that the E911 number defaults to the extension number.

Force Login: Determines whether or not login is forced at the extension. It is recommended that this field be left checked.

Media Control: Determines whether or not you want to control the media at the extension and, if so, where the media streams are to be routed. The available Media Control modes are discussed in detail below.

Dependency Mode: Determines which device controls the extension's physical elements and media. The available Dependency Modes are discussed in detail below.

Media Control Modes

DMCC devices can be registered in one of the following Media Control modes:

No Media: Select this mode if you want the DMCC Dashboard to be able to control and monitor the physical aspects of the extension, and for first-party call control. This mode is also sometimes referred to as "Shared Control" because it allows a client application to share control of an extension with the physical device or IP softphone logged into the extension. When you register a DMCC device in this mode, the fields under the **Phone Command** tab can be used to monitor and control the extension. See *Chapter 8: First-party Device Control* for more information.

Telecommuter: Select this mode to route all media to another phone set. When registering a DMCC device in this mode, you must enter the telephone number of the other phone set in the **Telecommute Number** field. The Telecommute Number is the dial string used to access the phone set. For example, if your home phone number is 20888888 and you dial 9 to get an outside line, you would enter "920888888" to route all incoming calls to your home.

Server Mode: Select this mode if you want the media at the extension to be routed to, and controlled by, the AE Services server. When you register a DMCC device in this mode, the fields under the **Server Media** tab can be used to collect tones, record calls, play messages, etc. See *Chapter 6: Server-side Media Control* for more information about using Server Media. This mode is suitable for demonstrating call recording and other media control functionality on the DMCC Dashboard, but does not provide the scalability or performance required for production-level applications.

Client Mode: Select this mode if you want the media at the extension to be controlled by the client application. The DMCC Dashboard simply routes the media to a specified IP address: in practice, your application would access this IP address and provide logic to manage the media streams, as required. When you register a DMCC device in this mode, the fields under the **Client Media** tab can be used to view details of the media streams from the extension and to exercise media redirection functionality. See *Chapter 7: Client-side Media Control* for more information.

Registering DMCC Devices in Client Media Mode

Figure 4-14 shows the additional fields that need to be considered when registering a DMCC device in Client Media mode.

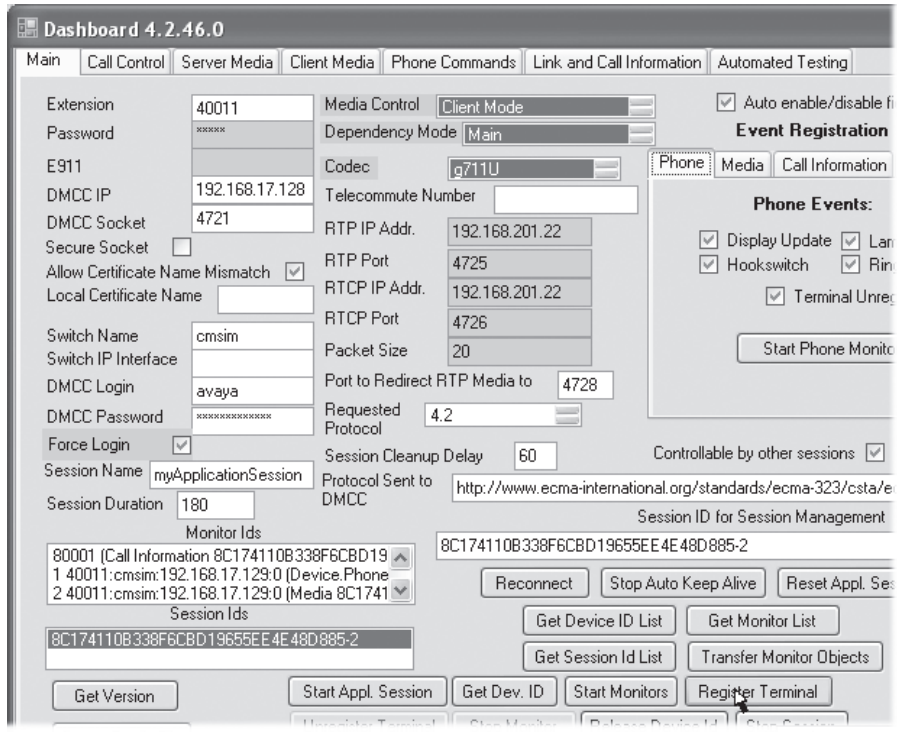

Figure 4-14: Registering DMCC Devices in Client Media Mode

The additional highlighted fields that need to be considered are:

Codec: A codec is an algorithm used to encode and decode audio media flowing from and to the DMCC device. At registration, the application can specify the list of codecs it supports. Communication Manager maintains separate lists of codecs supported for each type of device in each Network Region. Communication Manager determines the actual codec that is used, selecting a codec that is common to both lists. If there are no common entries, Communication Manager selects a null codec causing no media to be sent or received.

The DMCC Dashboard user interface only allows one supported codec to be selected when registering a DMCC device. In Client Media mode you can select from the supported codecs, namely G.729, G.729A, G.711A, G.711U or G.723. In other media modes, the codec defaults to G.711U and cannot be changed.

RTP IP Addr.: Defines the IP address to which Communication Manager will send Real Time Protocol (RTP) data at the extension. RTP is the protocol used to transport media data. The first time you start the DMCC Dashboard, the field defaults to the IP address of the machine on which the DMCC Dashboard is running. If your machine has two or more Network Interface Cards, there is no guarantee which one will be selected. The tool tip displayed when you place your mouse over the field lists the possible valid values.

RTP Port: The port number at the RTP IP address, to which RTP data at the extension will be sent. The default value is 4725.

RTCP IP Addr.: Defines the IP address to which Communication Manager will send Real Time Control Protocol (RTCP) data. RTCP provides control information for an RTP session. The default value is the IP address of the machine on which the DMCC Dashboard is running.

RTCP Port: The port number corresponding to the RTCP IP Address, to which RTCP data will be sent. The default value is 4726.

Packet Size: Determines the size, in milliseconds, of packets exchanged between the DMCC Dashboard and Communication Manager. Supported values are 10, 20, 30, 40, 50, and 60, although 20 or 30 are usual. If the field is left blank or set to 0, Communication Manager determines the packet size to be used.

Port to Redirect RTP Media to: When the DMCC Dashboard opens up media processing threads, the thread that sends out the media will be on this port. The default value is 4728 and need not normally be changed.

Dependency Modes

When the DMCC Dashboard, or any client application, registers itself as a DMCC device, it must specify its dependency mode. The dependency mode determines which device controls the extension's physical elements and media. The possible dependency modes are:

Main: Only one device can be registered at an extension with dependency mode Main. The device can originate and receive calls, can talk and listen, and block the transfer of talk capabilities to other devices registered at the extension. Typically, a physical set or IP softphone is registered as the Main device at an extension. However, a client application can register itself as the Main device at an extension, for example, if it wants to add the extension to calls at another extension so that it can obtain the media streams.

Independent: A device registered in this mode can originate and receive calls, and can talk and listen even if a Main device is not registered.

Dependent: A device can only be registered in this mode if a Main device has already been registered at the extension. Client applications are typically registered in this mode allowing them to share control of the physical aspects of the extension with the Main device, or to access the media streams at the extension.

Using Registered DMCC Devices

Once you have registered a DMCC device against an extension, you can use it to exercise the DMCC service's device and media control capabilities at the appropriate tabs on the DMCC Dashboard. The chapters that follow take a detailed look at these tabs and the capabilities they provide. You can also use the DMCC Dashboard IP Softphone to take control of the extension and make calls using first-party call control—see *Using the DMCC Dashboard IP Softphone*, below.

Unregistering DMCC Devices

Once a client application finishes controlling an extension, it should perform a clean-up operation and unregister the DMCC device. If the application fails to unregister a DMCC device, Communication Manager will keep it registered indefinitely.

To demonstrate unregistering a DMCC device on the DMCC Dashboard:

1. Select the Device ID in the **Device IDs** list box.
2. Click **Unregister Terminal**.

An `UnregisterTerminalRequest` message is sent to the AE Services server.

Converting Telephone Number Formats

The DMCC Dashboard allows you to exercise functionality that converts E.164 format telephone numbers to Communication Manager extension numbers, and vice versa.

E.164 is the international public telecommunication numbering format, used in the PSTN and other data networks. E.164 numbers have a maximum of 15 digits and are usually written with a + prefix.

Organizations often use the E.164 format to store their employees' telephone numbers in a directory. Dial Plan rules are used to map the E.164 numbers to the extension numbers provisioned on Communication Manager.

When an application performs a look-up based on an employee's name in an external directory, it may receive an E.164 format number. If the application wants to get a Device ID, it must first convert the E.164 number to a Communication Manager extension.

Conversely, if the application needs to retrieve the name of the employee at a particular extension, it may need to convert the extension number to an E.164 format number before performing the look-up in an external directory.

Converting E.164 Telephone Numbers

To convert E.164 telephone numbers to extension numbers, from the DMCC Dashboard:

1. Type the E.164 format numbers, one per line, in the large scrollable text box at the bottom of the **Main** tab, as shown in **Figure 4-15**.
2. Click **Convert E164 to Dial String**.

A `ConvertE164ToDialString` message is sent to the DMCC service which responds with a `ConvertE164ToDialStringResponse` message from which the extension number can be retrieved.

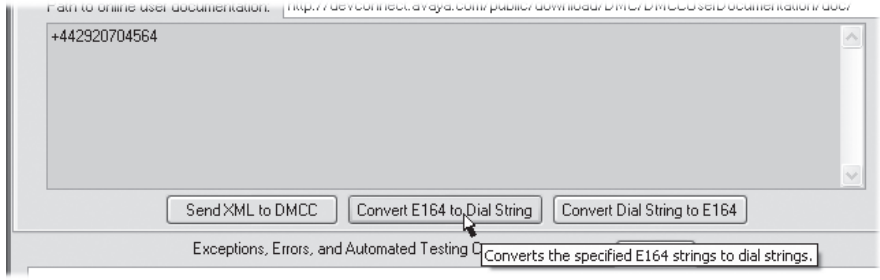

Figure 4-15: Converting E.164 Telephone Numbers

Converting Extension Numbers

To convert Communication Manager extension numbers to E.164 format telephone numbers, from the DMCC Dashboard:

1. Type the extension numbers in the text box, one per line.
2. Click **Convert Dial String to E.164.**

A `ConvertDialStringToE164` message is send to the DMCC service which responds with a `ConvertDialStringToE164Response` message from which the E.164 format numbers can be retrieved.

Using the Dashboard IP Softphone

When you register the DMCC Dashboard as a DMCC device at an extension, you can use the DMCC Dashboard IP Softphone to perform simple first-party call control at the extension, including making and receiving calls.

Note: You can use the IP softphone to control an extension, regardless of which media or dependency mode the DMCC Device is registered in.

In the examples in this section, the DMCC Dashboard is registered as a DMCC device at extension 40010, with all monitors started, the media mode set to “No media” and dependency mode set to “Dependent”. A virtual DCP phone set is the Main device at the extension; the DMCC Dashboard effectively shares control of the extension with the DCP phone set.

Controlling an Extension

To use the DMCC Dashboard IP Softphone to control an extension, click on the extension’s Device ID to highlight it in the **Device IDs** list box.

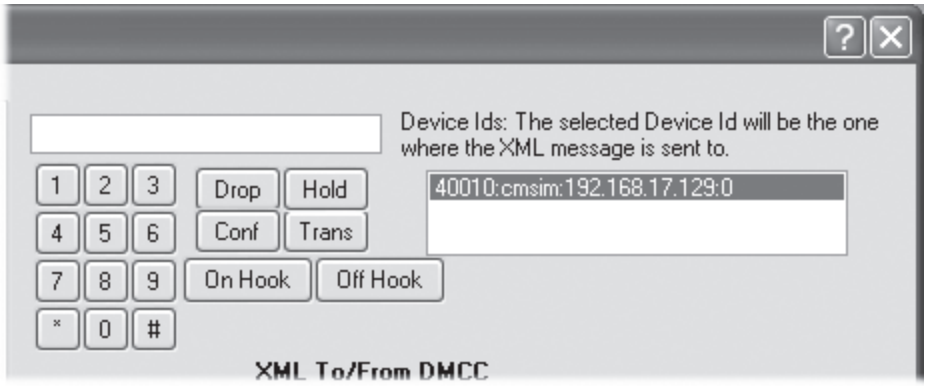

Figure 4-16: Selecting an Extension to Control

The DMCC Dashboard IP softphone controls become enabled.

Dashboard IP Softphone Options

The DMCC Dashboard IP softphone allows you to emulate performing the following tasks on the phone set at the controlled extension:

- **Off Hook**: Takes the handset off the switchhook, usually to answer or start making a call.
- **On Hook**: Puts the handset on the switchhook, usually to hang up all existing calls.
- **Trans**: Presses the Transfer button on the phone set, usually in preparation for transferring an existing call to another extension.
- **Conf**: Presses the Conference button on the phone set, usually in preparation for initiating or terminating a conference call.
- **Hold**: Presses the Hold button on the phone set, placing an existing call on hold.
- **Drop**: Presses the Drop button on the phone set, usually to drop out of an existing call.

The DMCC Dashboard IP Softphone also includes a keypad, for dialing telephone numbers, etc. and a display field, which shows the top line display at the controlled extension's phone set.

The DMCC Dashboard IP Softphone allows you to control a phone set's hookswitch status and press its fixed buttons. In addition to fixed buttons, a station can be provisioned with a wide variety of administered buttons whose functionality is assigned during station administration on Communication Manager. *Chapter 8: First-party Device Control* describes how to get information about administered buttons and their associated lamps, and

how to emulate pressing administered buttons to call their assigned functionality.

Making a Call Using the DMCC Dashboard IP Softphone

This example uses the registered DMCC device to control extension 40010 and make a call to extension 40011:

1. Click on extension 40010's Device ID to highlight it in the **Device IDs** list box.
2. Click **Off Hook**—a `SetHookSwitchStatus` message is sent to the DMCC service, instructing it to set extension 40010's Device ID hookswitch status to "false".
3. Click **4, 0, 0, 1, 1** on the keypad—a `ButtonPress` message is sent to the DMCC service for each key press, instructing it to emulate pressing the corresponding key at extension 40010.

As you perform these tasks on the IP softphone, the virtual DCP phone set at extension 40010 updates automatically, as shown in **Figure 4-17.** Responses and events from the DMCC service are displayed in the XML Messages and Events regions of the DMCC Dashboard as the call is dialed and established.

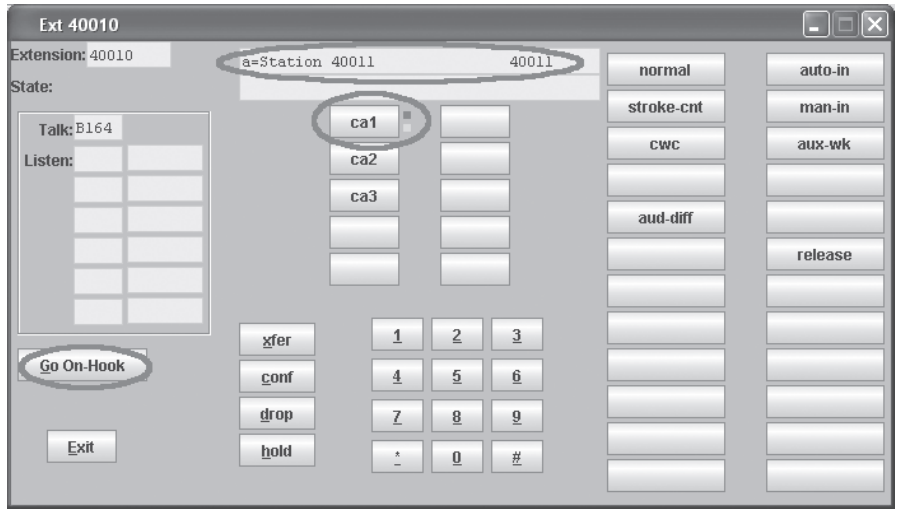

Figure 4-17: The DCP Phone at Extension 40010

Notice that top line display on the DCP phone set is shown in the Display field on the IP softphone.

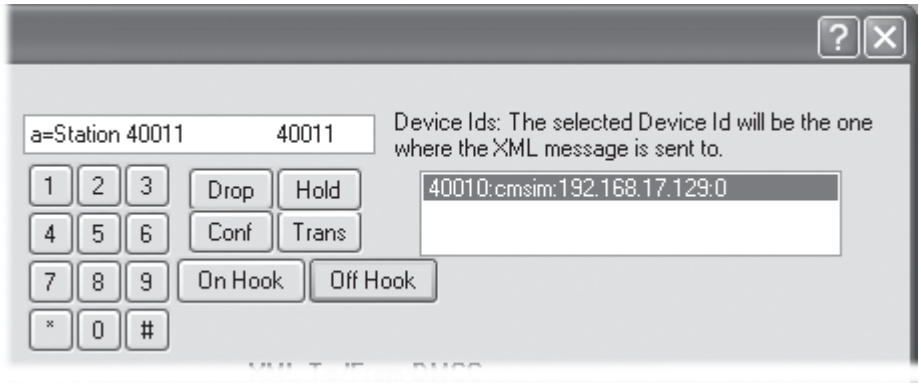

Figure 4-18: DMCC Dashboard IP Softphone on Call

The DCP phone at extension 40011 is now ringing, as shown in **Figure 4-19.**

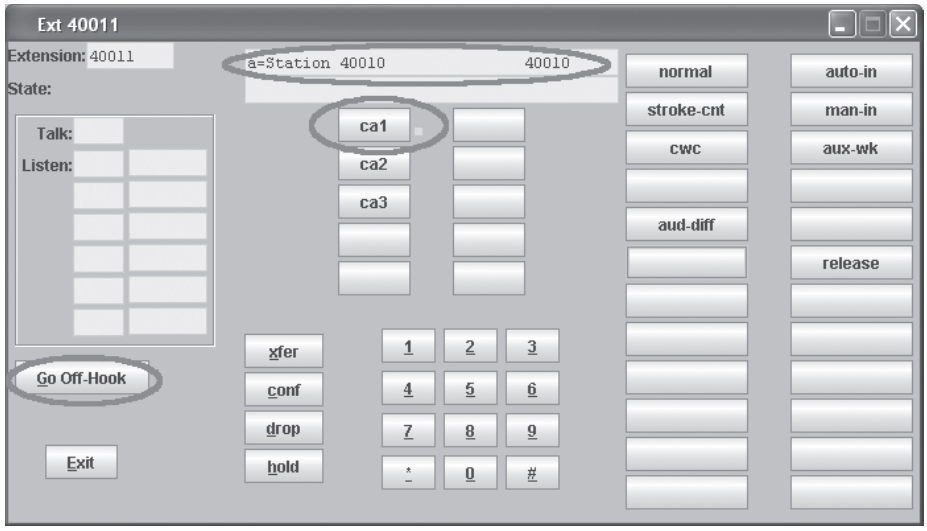

Figure 4-19: The DCP Phone at Extension 40010 in the Ringing State

Sending Raw XML Messages to the DMCC Service

Typically, when you select an option on the DMCC Dashboard, a corresponding .NET method is called. Using the associated field values you provide, the .NET method constructs an XML message describing your request, and sends it to DMCC service. Provided the **TO DMCC** check box is selected, the outgoing XML message is displayed in the **XML To/From DMCC** text box. In addition, provided the **FROM DMCC** check box is selected, the XML message responses from the DMCC service are displayed in the same field.

Being able to see the XML messages exchanged between the DMCC Dashboard and the DMCC service is of particular value to developers who want to create applications using the DMCC XML access method and SDK. XML developers can use the DMCC Dashboard to exercise required functionality and see both the outgoing XML they need their application to construct and the incoming XML they need their application to interpret.

In addition to calling .NET methods to construct XML messages, users can enter outgoing messages directly in the DMCC Dashboard, send them to the DMCC service and observe the results. XML developers can use this functionality to unit test their own raw XML, and to observe the effects of making manual changes to generated XML.

To send a raw XML message directly to the DMCC service:

1. Enter a correctly formatted XML message in the scrollable text box at the bottom of the **Main** tab. You can either type the XML in full or copy a previously generated outgoing message from the **XML To/From DMCC** text box. If copying a previously generated message, you can amend attribute values, as required. In **Figure 4-20**, the `GetDeviceId` message generated to get the Device ID for extension 40010 has been copied and amended to get the Device ID at extension 40011.
2. Click **Send XML to DMCC**. The message is sent to the DMCC service, and resultant response messages and events displayed in the XML Messages and Events region of the dashboard user interface.

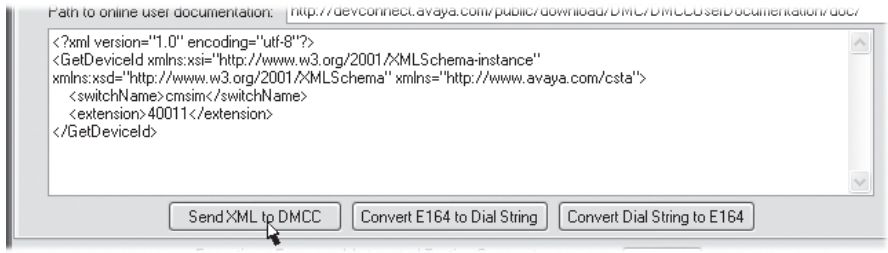

Figure 4-20: Sending XML Messages Direct to the DMCC Service

To clear the XML messages in this field, highlight all the text and delete it.

In future releases of the DMCC Dashboard (post 4.2 release) it will be possible to enter incoming XML messages into the DMCC Dashboard user interface. The dashboard processes the incoming messages as though they were sent by the DMCC service.

Configuring the DMCC Dashboard

When the DMCC Dashboard is first started, many of the fields are populated with default values. These defaults are loaded from the `config.xml` file located in the same installation directory as the DMCC Dashboard itself.

If you want to save different default values for any of these fields, or want to provide default values for fields that don't currently have them:

1. Enter the required default values at the **Main** and other tabs.
2. Click **Create Config File** on the **Main** tab, and confirm that you want to overwrite the existing `config.xml` file.

A new version of the `config.xml` file is created and will be used to load the default field values next time you start the DMCC Dashboard.

Using this feature saves you from having to remember and manually re-enter configuration settings each time you start the DMCC Dashboard.

You can save different sets of configuration settings in different files with different filenames. To apply a particular set of configuration settings, rename the saved file `"config.xml"`, copy it into the installation directory and start the DMCC Dashboard.

Other Main Tab Functionality

The **Main** tab includes a few miscellaneous fields whose purpose and use haven't already been described:

- **Auto Enable/Disable Fields:** When selected, buttons and fields only become active and available for use when the DMCC Dashboard is in the appropriate state and all prerequisite tasks have been completed. If deselected, all fields and buttons are available at all times. However, options will fail, and errors will be returned by the DMCC service, if the DMCC Dashboard is not in the required state. Generally, it is recommended that you keep this option selected for demonstration purposes, although you may wish to deselect it to observe how certain errors are handled and reported by the DMCC service.
- **Requested Protocol:** Select and highlight the version of AE Services that you are running the DMCC Dashboard against. This automatically updates the value in the **Protocol Sent to DMCC** field to ensure that the supported version of the XML messaging protocol is used.

- **Path to online user documentation:** This fields defaults to the base URL of the .NET API documentation on the DevConnect web site, and should not need to be changed.

Remember, to obtain on-line help and .NET API documentation for options on the DMCC Dashboard, click on the **?** icon in the top right-hand corner of the DMCC Dashboard window and then click on the corresponding control.

Round-up

You are now familiar with the options on the DMCC Dashboard **Main** tab and have an understanding of how the functionality exercised by these options should be used in your own DMCC client applications. In particular, you have learned how to: start and manage application sessions, get first-party Device IDs, start event monitors, register DMCC devices, use the DMCC Dashboard IP softphone, send raw XML messages directly to the DMCC Service and configure the DMCC Dashboard.

In the chapters that follow you will learn how to use the DMCC Dashboard to exercise the DMCC service's third-party call control, server- and client-side media control, and first-party device control capabilities. Let's start by looking at the third-party call control options available under the **Call Control** tab.

Chapter 5

Third-party Call Control

In this chapter:

- Introducing the third-party call control, snapshot and logical device services
- Finding out about the device, call and connection identifiers used for third-party call control
- Exploring the DMCC Dashboard **Call Control** tab
- Getting ready to exercise call control capabilities
- Establishing calls using third-party call control
- Exercising the DMCC service's other third-party call control capabilities

As you learned in *Chapter 1*, the "Call Control" element of Device, Media and Call Control refers to the basic third-party call control capabilities provided by the AE Services DMCC service. Client applications can use these capabilities to create and manipulate calls remotely. For example, to make a call from one extension to another, the client application issues a simple, high-level request.

In addition to third-party call control, the DMCC service provides:

- **Logical Device** capabilities that allow applications to set call forwarding and "do not disturb" notices at extensions.
- **Snapshot** capabilities that allow applications to obtain information about specified calls or all calls at specified extensions.

This chapter looks at the DMCC service's third-party call control, snapshot and logical device capabilities and describes how the DMCC Dashboard can be used to exercise, observe and monitor them.

The options used to access these capabilities are grouped under the **Call Control** tab in the DMCC Service Options region of the DMCC Dashboard user interface.

Identifying Extensions, Calls and Connections

Many of the third-party call control options require you to identify the extensions, calls and/or connections that you want to manipulate.

Third-party Device IDs

Up until now in this book, first-party Device IDs have been used to identify extensions at which you want to start event monitors and perform first-party device or media control. However, a different type of Device ID is used to identify extensions for third-party call control. Third-party Device IDs differ from first-party Device IDs in that:

- They are not exclusive to an application session.
- They do not need to be explicitly released when an application session is cleaned up.

First-party Device IDs can be used for third-party call control provided they include the switch connection name. The switch connection name is included only if there is an H.323 link between the gatekeeper and the call controller.

In fact, the first-party Device IDs used so far do include the AE Services switch connection name ("cmsim"), so they could be used for third-party call control. However, this may not always be the case, so third-party Device IDs will be used for the examples in this chapter. See *Making and Answering Calls* below for information about how to get the third-party Device IDs of extensions.

Call IDs

When a call is originated it is automatically assigned a Call ID. Many of the third-party call control options act upon existing calls and require Call IDs to be specified.

When you use third-party call control to make a call, the Call ID can be obtained from the resultant response message or associated event. However, if a call is established by some other method, such as manually or using first-party call control, you can use the DMCC service's snapshot capabilities to get its Call ID. See *Making and Answering Calls* below for information about how to get Call IDs.

Connection IDs

A "connection" can be considered to be a single leg of a call, corresponding to a particular extension's involvement in a particular call.

Each connection in a call is identified by a unique Connection ID, made up of the extension's third-party Device ID and the Call ID. Many of the call control options require you to provide one or more Connection IDs to identify the connections and calls that you want to control.

The Call Control Tab

The **Call Control** tab on the DMCC Dashboard comprises the controls used to exercise all of the DMCC service's third-party call control capabilities.

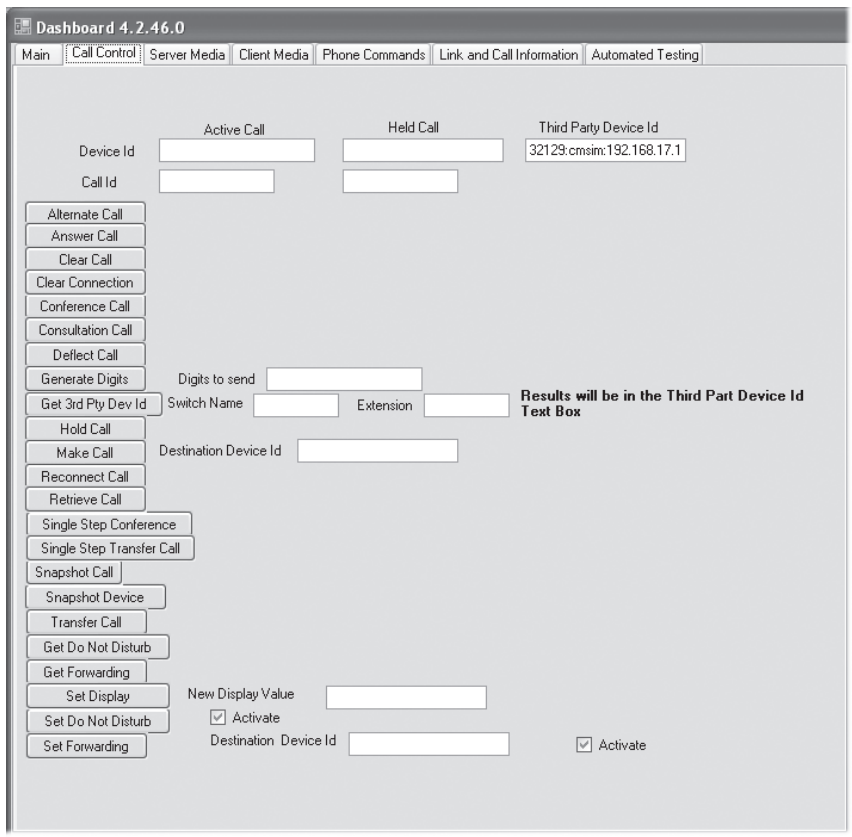

Figure 5-1: The **Call Control** Tab

The available call control options are laid out from top to bottom of the tab. What these options do and how they are used is described later in this chapter, including an example showing how to establish a call.

The **Answer Call, Held Call** and **Third Party Device ID** fields across the top of the tab are used to specify the extensions, calls and connections you want to control.

In future releases of AE Services (post 4.2 release), the DMCC service will support additional third-party call control capabilities. The DMCC Dashboard will be updated to support these new capabilities and the corresponding options made available at the **Call Control** tab.

In future releases of the DMCC Dashboard (post 4.2 release) the look and feel of the **Call Control** tab will be significantly enhanced. In particular, the fields associated with a particular option will be automatically displayed when users place their mouse over the corresponding button. The new design will make it easier for users to identify which fields need to be filled in for each option.

Preliminary Steps

Before you can use the DMCC Dashboard to demonstrate the call control options, you must perform the following tasks:

1. Start an application session. See *Chapter 4: Starting Application Sessions.*
2. Get the first-party Device IDs of the extensions you want to control. See *Chapter 4: Getting Device IDs.* Getting first-party Device IDs is not essential for third-party call control, but is required if you want to start monitors at the extensions.
3. Set Call Control and other required monitors at the extensions you want to control—see *Chapter 4: Starting and Managing Monitors.* Setting monitors is not mandatory for third-party call control. However, to understand how third-party call control works it is important to see the call control events that are generated. In addition, any third-party call control applications you create will need to monitor Call Control events and implement the corresponding callback methods.

DMCC device registration is **not** required for third-party call control.

The examples in this chapter use Avaya IPCoDE DCP extensions 40010, 40011 and 40012, and an Avaya IP Softphone logged into extension 32129. The first-party Device IDs of the extensions have been retrieved and call control monitors set at each of the extensions.

Once an application session is started, and monitors set, the **Call Control** tab can be used to exercise the DMCC service's call control, snapshot and logical device capabilities.

Making and Answering Calls

Perhaps the most ubiquitous third-party call control task involves establishing a call between two extensions. Once a call is established, other third-party call control options can be used manipulate it.

The example below describes how a simple, two-party call can be successfully established using third-party call control. Other outcomes are possible if, for example, the destination extension is unavailable, or call forwarding is set. Establishing the call involves:

- Getting the third-party Device IDs of the extensions you want to put into the call.
- Making the call from one extension to the other.
- Getting the Call ID of the call.
- Answering the call at the destination extension.

Getting Third-party Device IDs

To get the third-party Device ID for an extension, from the **Call Control** tab:

1. Enter the **Switch Name** and **Extension** number in the appropriate fields.
2. Click **Get 3rd Pty Dev Id**.

A `GetThirdPartyDeviceId` message is sent to the DMCC service. The result is displayed in the **Third-party Device ID** field at the top of the tab.

Figure 5-2: Getting Third-party Device IDs

Figure 5-3: Get Third-party Device ID Result

The format of the third-party Device ID includes the switch connection name. Currently the switch name is case sensitive. For example, Device ID "32129:cmsim::0" is not the same as "32129:CmSim::0". This distinction may be removed in later AE Services releases.

The third-party Device IDs of the extensions to be put into the call are:

32129:cmsim::0

40010:cmsim::0

Making Calls

To make a call from extension 32129 to extension 40010:

1. In the **Third-party Device ID** field, enter the Device ID for extension 32129.
2. In the **Destination Device ID** field, enter the Device ID for extension 40010.
3. Click **Make Call**.

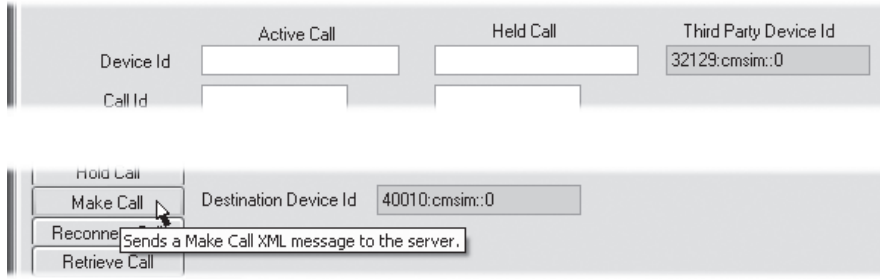

Figure 5-4: Making a Call

A `MakeCall` XML message is sent to the server. Extension 32129 goes off hook and calls extension 40010, which remains in the ringing state until it is answered.

Assuming that Call Control event monitors have been started at both extensions, the following message and events are received at the DMCC Dashboard:

- `MakeCallResponse` message
- `Originated` event, indicating that an attempt has been made to make a call from extension 32129
- `Delivered` event monitored at both extensions, indicating that a call has been presented to extension 40010 in ringing mode

The `MakeCallResponse` message, and each of the associated events, include the Call ID assigned to the call. The message and event details are displayed in the **Events** text box in the Messages and Events region of the DMCC Dashboard user interface.

```
Make Call Response: 55
   CallingDeviceConnectionId: Device Id: 32129:CMSIM::0 Call
Id:3590
Originated Event: 1
   Originated Connection Id: Device Id: 32129:CMSIM::0 Call
      Id:3590
   Calling Device Id: 32129:cmsim:192.168.17.129:0
   Called Device Id: 40010:CMSIM::0
   Local Connection Info.: initiated
   Event Cause: newCall
Delivered Event: 2
   Connection Id: Device Id: 40010:CMSIM::0 Call Id:3590
   Alerting Device Id: 40010:cmsim:192.168.17.129:0
   Calling Device Id: 32129:CMSIM::0
   Called Device Id: 40010:cmsim:192.168.17.129:0
   Last Redirection Device Id: 40010:cmsim:192.168.17.129:0
   Local Connection Info:alerting
   Event Cause: newCall
Delivered Event: 1
   Connection Id: Device Id: 40010:CMSIM::0 Call Id:3590
   Alerting Device Id: 40010:CMSIM::0
   Calling Device Id: 32129:cmsim:192.168.17.129:0
   Called Device Id: 40010:CMSIM::0
   Last Redirection Device Id: 40010:CMSIM::0
   Local Connection Info:connected
   Event Cause: newCall
```

Getting Call IDs

Because the **Make Call** option was used to make the call, the Call ID could be retrieved from the resultant response message or events. However, if the call was made by another method, such as manual dialing or using the DMCC Dashboard IP softphone, you may not have access to these messages and events. An alternative way of retrieving the Call ID of a call you want to answer or control is to use the DMCC service's snapshot capabilities. To get the Call IDs of all calls at a particular extension:

1. Enter the extension's third-party Device ID in the **Third-party Device ID** field.
2. Click **Snapshot Device**. A `SnapshotDevice` message is sent to the server.

The server returns a `SnapshotDeviceResponse` message that includes the Call IDs and connection states of all calls at the extension. The full XML message is displayed in the **XML To/From DMCC** text box and extracted details appear in the **Events** text box. For example, the following call details are displayed in the **Events** text box for extension 32129:

```
Device Id:32129:CMSIM::0
  Call Id:7216
  Local Connection State connected
  Local Connection State connected
Device Id:32129:CMSIM::0
  Call Id:6371
  Local Connection State hold
  Local Connection State connected
```

In this case, there are two calls at the extension, one active and one on hold.

To make life easier for users, the DMCC Dashboard automatically populates the **Active Call** and **Held Call** fields with details of the calls. Alternatively, you can copy or transcribe the Call ID from the response into the appropriate fields. Obviously, in your own applications, you will have to get the Call ID from the response message programmatically.

Answering Calls

After using the **Make Call** option, the destination extension is ringing. To answer the call:

1. Enter extension 40010's Device ID in the **Active Call**, **Device ID** field.
2. Enter the Call ID in the **Active Call, Call ID** field.
3. Click **Answer Call**.

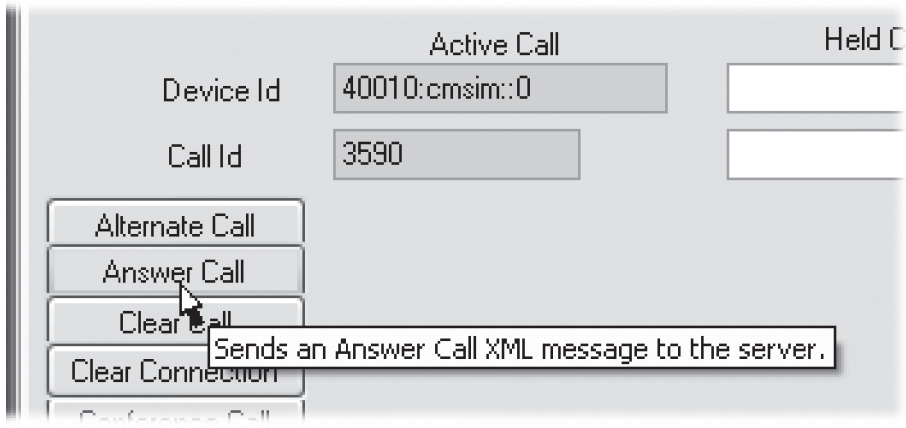

Figure 5-5: Answering a Call

An `AnswerCall` message is sent to the server. Extension 40010 goes off hook and the call is established. An `AnswerCallResponse` message and `Established` event are generated.

```
Events:
  Established Event: 2
   Connection Id: Device Id: 32129:CMSIM::0 Call Id:3590
   Alerting Device Id: 32129:CMSIM::0
   Calling Device Id: 32129:CMSIM::0
   Called Device Id: 40010:cmsim:192.168.17.129:0
   Last Redirection Device Id:
   Event Cause: newCall
Answer Call Response: 56
```

Answer Call can be used to answer any call offered at an extension, regardless of how it was made: whether it was using the **Make Call** option, first-party-call control or by manual dialing.

Third-party Call Control Options

Now that you have learned how to establish calls, you can go on to exercise other third-party call control capabilities that allow you to manipulate those calls. The DMCC Dashboard's tooltips, on-line help and user guide provide detailed instructions on how to use the various options on the **Call Control** tab, so they aren't all demonstrated in this book. However, it is worth finding out what the available options are and what they will allow you to do in your own applications.

The tables on the pages that follow list the third-party call control, snapshot, logical device and other options available on the **Call Control** tab. For some options, an example is given to help explain their behavior.

Table 5-1
Third-party Control Options

Alternate Call	If there is an active call and one or more held calls at an extension, the active call is put on hold and the specified held call made active. When demonstrating this option on the DMCC Dashboard, the Device IDs in the **Active Call** and **Held Call** fields are the same because the active and held calls are at the same extension. However, the Call IDs are different. **Example**: extension 32129 is in an active call with extension 40010 (Call ID 7216) and has a call with extension 40011 on hold (Call ID 6371). Using this option puts call 7216 on hold and makes call 6371 active.
Answer Call	If a call is ringing or being offered to an extension, the call is answered. On the DMCC Dashboard, the call is identified by the **Device ID** and **Call ID** of the extension's connection, in the **Active Call** field.
Clear Call	Releases all connections from a specified call. On the DMCC Dashboard, the call is identified by the **Device ID** and **Call ID** of any one of its connections, in the **Active Call** field.
Clear Connection	Releases the extension specified in the **Active Call**, **Device ID** field from the call specified in the **Active Call**, **Call ID** field.
Conference Call	If there is an active call and one or more held calls at an extension, the option merges the active and specified held call into a conference call. As for the Alternate Call option, the Device IDs in the **Active Call** and **Held Call** fields are the same, but the Call IDs are different.
Consultation Call	If there is an active call at an extension, it is put on hold and an outgoing call made to the extension specified in the **Third Party Device ID** field.
Deflect Call	If an incoming call (specified in the **Active Call** fields) is not yet established, this option redirects the call to the extension specified in the **Third Party Device ID** field.
Generate Digits	Sends DTMF or rotary digits, entered in the **Digits to send** field, to all other connections in a specified call as though from the connection specified in the **Active Call** fields.
Hold Call	If there is an active call at an extension (specified in the **Active Call** fields) it is placed on hold.

Table 5-1
Third-party Control Options Cont.

Make Call	Initiates a call from the extension specified in the **Third Party Device ID** field to the extension specified in the **Destination Device ID** field.
Reconnect Call	If there are held calls at an extension, this option makes the specified held call active. If there was an active call at the extension when the option was selected, it is either cleared or put on hold depending on the station settings. When demonstrating this option, the Device IDs in the **Active Call** and **Held Call** fields are the same but the Call IDs are different.
Retrieve Call	If there are held calls at an extension, the held call specified and the **Held Call** field is made active.
Single Step Conference Call	The option adds the extension specified in the **Third Party Device ID** field to an existing call specified in the **Active Call** fields. The new extension is forced off hook on speakerphone. Unlike the **Conference Call** option, there is no point at which there are two calls so there is no merging, the new extension never goes into ringing state and there is no opportunity for consultation.
Single Step Transfer Call	Transfers the call specified in the **Active Calls** fields to the extension specified in the **Third Party Device ID** field. The extension the call is transferred to is forced off hook on speakerphone. Unlike the **Transfer Call** option, there is no point at which there are two calls and there is no opportunity for consultation.
Transfer Call	If there is an active and one or more held calls at an extension, this option transfers the specified held call to the destination extension of the active call. **Example:** Extension 32129 is in an active call with extension 40010 (Call ID 7217). To begin transferring the call, the connection to extension 40010 is put on hold and a call made from extension 32129 to extension 40012 (Call ID 6372). When the **Transfer Call** option is used, call 7217 is transferred from extension 32129 to extension 40012, so that extensions 40010 and 40012 are connected in a call.

Table 5-2
Snapshot Options

Snapshot Call	Returns information about the extensions participating in the call that includes the connection specified in the **Active Call** fields.
Snapshot Device	Provides information about calls at the extension specified in the **Third Party Device ID** field. For convenience, the DMCC Dashboard automatically populates the **Active Call** and **Held Call** fields with the call details returned.

Table 5-3
Logical Device Options

Get Do Not Disturb	Returns the "Do Not Disturb" state of the extension specified in the **Third Party Device ID** field. "Do Not Disturb" equates to the Communication Manager feature "Send All Calls (SAC)".
Get Forwarding	Returns the forwarding state of the extension specified in the **Third Party Device ID** field.
Set Do Not Disturb	If **Activate** is checked, "Do Not Disturb" is set on the extension specified in the **Third Party Device ID** field. The extension will not be alerted of any incoming calls whilst "Do Not Disturb" is set. If **Activate** is not checked, "Do Not Disturb" is removed from the extension.
Set Forwarding	If **Activate** is checked, the option causes all future calls to the extension specified in the **Third Party Device ID** field to be forwarded to the extension specified in the **Destination Device ID** field. If **Activate** is not checked, the option turns off call forwarding at the specified extension.

Table 5-4
Other Call Control Tab Option

Get 3rd Pty Dev ID	Returns the third-party Device ID of an extension defined by its **Switch Name** and **Extension** number, and displays it in the **Third Party Device ID** field.
Set Display	Writes the text in the **New Display Value** field to the display on the phone at the specified extension. Note that this option only works if the AE Services server has been configured to support it.

Round-up

In this chapter you were introduced to the DMCC service's third-party call control, snapshot and logical device capabilities. The DMCC Dashboard **Call Control** tab was explored and used to establish a call. Finally, the full range of available third-party call control options was described.

In the next couple of chapters, you will learn about the DMCC service's media control capabilities, starting with server-side media control where AE Services provides the required media processing resources.

Chapter 6

Server-side Media Control

In this chapter:

- Introducing server-side media control
- Exploring the **Server Media** tab
- Playing messages to call participants
- Recording calls to WAV files
- Dubbing recorded calls
- Detecting and collecting DTMF tones

To access and control the media streams at an extension, a client application has to register itself as a DMCC device at that extension. The application can choose to:

- Process the media itself, in which case it registers in Client Media mode. Client-side media processing is described in *Chapter 7: Client-side Media Control.*
- Use AE Services to process the media on the server-side, in which it registers in Server Media mode. Server-side media processing is described in this chapter.

In Server Media mode, the media streams from the DMCC device terminate at the AE Services server. AE Services provides media control resources that allow client applications to:

- Record calls
- Play prerecorded messages to the participants in calls
- Insert prerecorded messages into call recordings
- Detect and retrieve telephone keypad presses (DTMF tones)

This chapter takes a detailed look at the DMCC service's server-side media control capabilities and how they can be demonstrated using the DMCC Dashboard.

Although suitable for small applications and for demonstration purposes, server-side media control does not provide the scalability or performance required by enterprise-level applications; client-side media control should be used instead.

About Server-Side Media Control

When a call is established, the summed media from each of the participants in the call is available at each of the extensions involved in the call. Thus, when a DMCC device is registered at an extension it is able to access the media for all calls that the extension is involved in. When a DMCC device is registered in Server Media mode, Communication Manager directs the media streams at the extension to the AE Services server. Server-side media control allows you to take advantage of AE Services media control resources to record calls, play messages and collect DTMF tones.

Server Media Mode RTP Media Parameters

In Client Media mode, the RTP media parameters, which determine how media data is transported and where it terminates, are set by the client application when the DMCC device is registered. In Server Media mode, the RTP media parameters are predefined and cannot be amended by the client application. The RTP media parameters for server-side media control are:

- **Codec:** G.771U. A codec is an algorithm used to encode and decode audio media flowing from and to the DMCC device. G.711U is an uncompressed codec that the server converts to PCM (pulse-code modulation) for saving or playing as a WAV file.
- **RTP IP Address and Port Number**: The IP address and port number where the media control resources are located on the AE Services server. RTP is the protocol used to transport media data.

- **RTCP IP Address and Port Number:** The IP address and port number where the media control resources are located on the AE Services. RTCP is the protocol used to provide control information about the RTP data.
- **Packet Size:** 20 milliseconds.
- **Media Encryption:** None.

Media Events

Client applications that use AE Services to provide media control need to be informed of Media events at the extensions they are controlling. For example, if an application wishes to play a message continuously to callers until they press the # key, it will need to listen for a `Tones Detected` event. In the associated callback method, the client application checks which key was pressed and, if #, stops the message.

Media event monitors listen for the following events:

- **Media Started Event:** A `Media Started` event is generated whenever a new RTP session is started at the monitored extension. When a call is established at a monitored extension which has a Server Media mode DMCC device registered against it, a `Media Started` event is generated containing the Server Media mode RTP media parameters described above.
- **Media Stopped Event:** A `Media Stopped` event is generated whenever an RTP session is disconnected at the monitored extension.
- **Playing Event:** A message is being played on the monitored extension's voice unit. The event includes the name of the file being played.
- **Playing Stopped Event:** A message that was being played at the monitored extension has stopped. The event includes the name of the message file and the reason for it being stopped.
- **Suspend Playing Event:** A message being played at the monitored extension has been suspended.
- **Tones Retrieved:** The monitored extension has retrieved a series of DTMF digits from the buffer. The event includes the list of keypad presses and the reason for them being retrieved.
- **Recording Message Event:** A call is being recorded at the monitored extension. The event includes the name of the recording file and the time the recording started or resumed.

- **Recording Stopped Event:** The recording of a call at the monitored extension has stopped. The event includes the name of the file and the time recording was stopped.
- **Suspend Recording Event**: A recording of a call at the monitored extension has been suspended. This could be to allow a prerecorded message to be dubbed into the recording.
- **Tone Detected Event:** The monitored extension has received a DTMF digit.

The Server Media Tab

The controls used to exercise the DMCC service's server-side media control capabilities are located on the DMCC Dashboard **Server Media** tab.

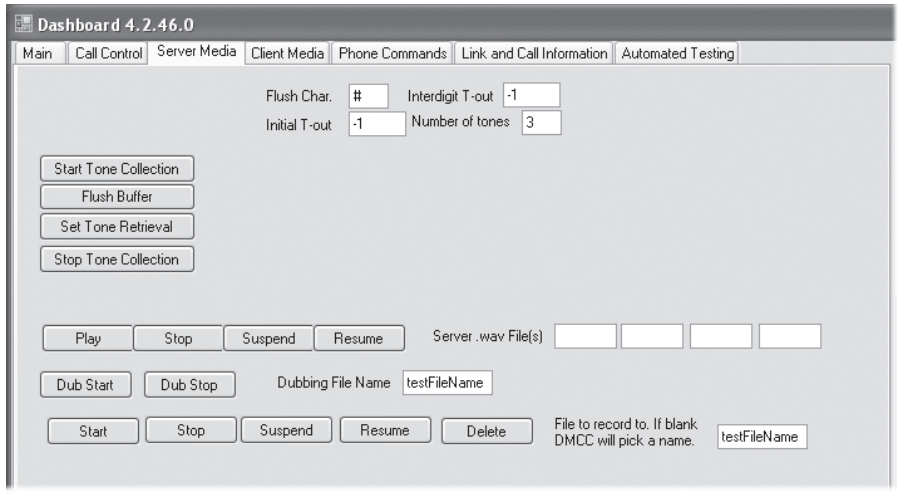

Figure 6-1: The **Server Media** Tab

The **Server Media** tab incorporates controls that allow you to exercise the following server-side media control capabilities:

- Record media from a call into a WAV file. Note that the media stream at an extension comprises the combined media from all of participants in a call. Thus, applications only need to access the media stream at one extension to record a whole call.
- Dub a recording. That is, insert a pre-recorded message into a recording.
- Play one or more prerecorded messages or tones to the other participants on a call.
- Stop, pause and resume play and record operations.

Server-side Media Control Using Avaya IPCoDE

Up until this point, Avaya IPCoDE has provided the installations of Communication Manager and AE Services used in this book. A limitation of Avaya IPCoDE is that it does not provide media processing resources and therefore cannot be used to demonstrate server-side call recording or to play prerecorded messages. If you want to exercise these capabilities yourself, you will need to run the DMCC Dashboard against full installations of Communication Manager and AE Services. However, Avaya IPCoDE can be used to demonstrate DTMF tone detection and collection.

- Detect and collect DTMF tones.

These options are described in detail in the sections that follow.

Preliminary Steps

Before you can use the DMCC Dashboard to exercise the DMCC services' server-side media control capabilities, you must:

1. Start an application session. See *Chapter 4: Starting Application Session.*
2. Get the first-party Device ID of the extension whose media you want to control. The examples in this chapter control the media at extension 32129, which has an Avaya IP Softphone logged into it as the Main device. See *Chapter 4: Getting Device IDs.*
3. Start the Media and other required event monitors at the extension. See *Chapter 4: Starting and Managing Monitors.* Starting event monitors is not mandatory for server-side media control, but, to help understand how it works, it is useful to see the events that are generated. In addition, any media control application you create will need to listen for Media, Call Control and other events at the extensions it is controlling.
4. Register a dependent DMCC device, in Server Media mode, against the extension you want to control—for the examples, against extension 32129. See *Chapter 4: Registering DMCC Devices.*
5. Put the extension into a call. For the examples, make a call from

extension 32129 to any other extension.

You can now use the registered DMCC device to manage the RTP media stream at extension 32129.

Playing Prerecorded Messages

The DMCC service allows you to play one or more prerecorded messages or tones to the participants in an active call. The messages are played on the RTP streams of the other participants in the call. They appear to come from the registered extension although they actually come from the server. Messages can be played once, continuously, for a specified duration, or until a specified DTMF tone is detected at the extension.

Because playing messages is associated with an extension, rather than with a call, the messages may continue across multiple calls in which the extension is a participant.

The DMCC Dashboard allows you to play up to four prerecorded messages to the participants in an active call. The messages are played continuously until playing is explicitly stopped.

To exercise the message playing capabilities of the DMCC service, from the **Server Media** tab:

1. In the **Device IDs** list box select the Device ID for extension 32129. Remember, the DMCC Dashboard has already registered itself as a Server Media mode DMCC device at this extension, and the extension is in a call with DCP extension 40010.
2. Enter the names of up to four message files in the **Server .wav File(s)** field. These must be the names of existing WAV files located in the DMCC player directory configured on AE Services.
3. Click **Play**, as shown in **Figure 6-2**. A `PlayMessage` message is sent

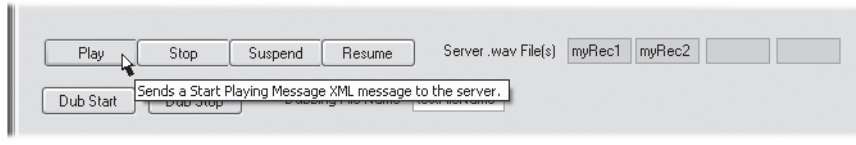

Figure 6-2: Playing Messages

the server. The message includes the selected extension's Device ID and the file names of the messages. A `Playing` event is generated.

To stop playing the message, click **Stop**. A `StopPlaying` message is sent to the server, which generates a `Stopped` event.

Whilst the messages are playing, you can **Suspend** them (that is pause or temporarily stop) and then **Resume** playing them.

Server-side Call Recording

To record an active call from the DMCC Dashboard **Server Media** tab:

1. Enter a name for the recording in the **File to record to** field. The recording will be saved in a WAV file with this name. If the field is left blank, the DMCC service will assign a name to the recording file.
2. In the **Device IDs** list box at the top-right of the dashboard, click on the Device ID of the extension at which the call is to be recorded. In this case, select extension 32129's Device ID.

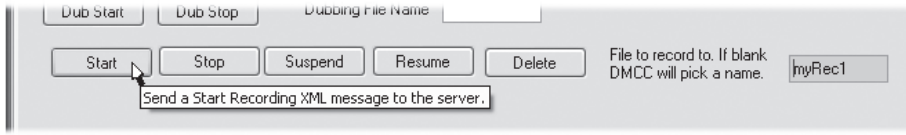

Figure 6-3: Starting Server-side Call Recording

3. Click **Start**, as shown in **Figure 6-3**. A `RecordMessage` message is sent to the DMCC service. The message includes the extension's Device ID and the recording file name. A `Recording` event is generated.
4. At the end of the call, or when you want to stop recording, select the extensions' Device ID in the **Device IDs** list box and press **Stop**. A `StopRecording` message is sent to the server. A `Recording Stopped` event is generated. The recording is saved in the DMCC recorder directory configured on AE Services.

You can also **Suspend** (that is, pause or temporarily stop) a recording in progress and then **Resume** the recording, or **Delete** a previously saved recording.

An error message will be returned if you attempt to stop, suspend, resume or delete a non-existent or invalid recording file.

Dubbing Call Recordings

The dubbing option allows you to insert a prerecorded message or tone into a call recording. When dubbing is started, the specified WAV file is played repeatedly over the recording until dubbing is explicitly stopped.

Dubbing can be used, for example, to avoid recording sensitive information such as a spoken password or other private or security-based information.

To dub a call recording that has already been started:

1. In the **Device IDs** list box, select the Device ID of the extension at which the call is being recorded.
2. Enter the **Dubbing File Name**. This must be the name of an existing WAV file located in the DMCC player directory configured on AE Services.
3. Click **Dub Start**, as shown in **Figure 6-4**. A `StartDubbing` message is sent to the server. The message includes the Device ID and the dubbing file name. The server suspends the call recording and a `Recording Suspended` event is generated.

Figure 6-4: Starting Call Recording Dubbing

4. To stop including the dubbing file and resume recording of the call, click **Dub Stop**. A `StopDubbing` message is sent to the server, which resumes the call recording. A `Recording` event is generated.

Detecting and Collecting DTMF Tones

Sometimes applications need to detect or collect DTMF tones arriving at a particular extension from other parties on a call. For example, the application may need to respond to a selected menu option, or be able to collect a customer's electricity meter reading entered via their telephone keypad.

DTMF tones are generated by participants pressing keypad characters **0** to **9**, * and **#** during a call. When a client application is registered as a Server Media mode DMCC device at an extension involved in a call, it can use the DMCC service to receive tones from other extensions in the call.

The DMCC service can be used in either of two ways:

- Each individual DTMF tone is reported to the client application as the keypad character is pressed. This is known as "tone detection."
- DTMF tones are accumulated and stored in a buffer. The tones in the buffer are retrieved by the client application when specified retrieval criteria are met. This is known as "tone collection."

The DMCC Dashboard can be used to demonstrate tone detection and tone collection.

Tone Detection

To demonstrate tone detection on the DMCC Dashboard:

1. Place a call from extension 32129 (which has a DMCC device registered against it in Server Media mode) to DCP extension 40010.
2. At extension 40010, press a series of characters on the virtual DCP phone's keypad.

For each keypad character, a separate `Tone Detected` event is generated and its details displayed in the Events text box in the Messages and Events region of the DMCC Dashboard. The event includes which keypad character was pressed.

Tone Collection

To exercise the DMCC service's tone collection capabilities, you must set criteria to determine when the buffered tones will be retrieved by the client, and then explicitly start tone collection.

Setting Tone Retrieval Criteria

When using tone collection, DTMF tones are buffered until specified retrieval criteria are met, at which point the tones are reported to the client application. The following retrieval criteria can be specified:

- **Flush Character:** The received tones are reported to the client application, and the buffer emptied ("flushed") when the specified flush character is detected.
- **Initial Timeout:** Defines the maximum period of time, in milliseconds, from tone collection being started until the first keypad character is pressed. If a tone is not received during this period, tone collection is stopped. "-1" indicates no initial timeout period.

- **Inter Digit Timeout:** Defines the maximum period of time, in milliseconds, between receiving consecutive DTMF tones. If, after receiving the previous tone, another tone is not received within this period, the tones in the buffer are reported to the client application, the buffer emptied and tone collection stopped. "-1" indicates no inter digit timeout period.
- **Number of Tones:** After the specified number of tones has been received in the buffer, they are reported to the client application and the buffer is emptied.

The buffered tones are reported to the client when the first of these criteria is met.

Collecting DTMF Tones

To demonstrate tone collection on the DMCC Dashboard:

1. Place a call from extension 32129 to 40010, as for *Tone Detection* above.
2. In the **Device IDs** list box, select and highlight the Device ID for extension 32129.
3. At the **Server Media** tab, enter the required flush character, initial timeout period, inter-digit timeout period and number of tones in the appropriate fields.
4. Press **Set Tone Retrieval**. A `ToneRetrievalCriteria` message is sent to the server.

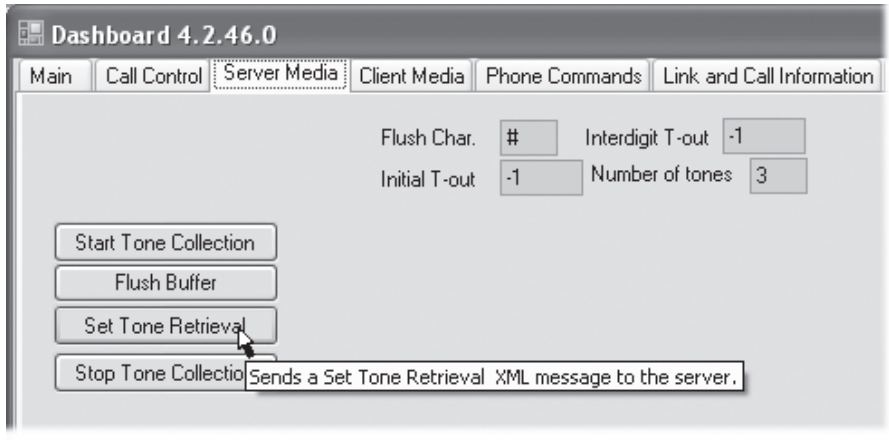

Figure 6-5: Setting Tone Retrieval Criteria

5. Click **Start Tone Collection**. A `ToneCollectionStart` message is sent to the server.
6. At extension 40010, press a series of characters on the virtual DCP phone's keypad.

When the first retrieval criterion is met (that is, when the flush character is pressed, or the specified number of tones is reached or a timeout occurs, whichever happens first) a `Tones Retrieved` event is generated and displayed in the **Events** text box. The events contain a list of the tones and details of the criterion that caused them to be retrieved.

It is also possible to empty the buffer without retrieving the tones by selecting the extension's Device ID and clicking **Flush Buffer**. A `ToneCollectionFlushBuffer` message is sent to the server.

To stop collecting tones at an extension, select the extension's Device ID and click **Stop Tone Collection**. A `StopToneCollection` message is sent to the server.

Round-up

In this chapter you were introduced to the DMCC service's server-side media control capabilities and learned how these can be demonstrated on the DMCC Dashboard. You can use these capabilities in your own applications if you want AE Services to process media streams.

In the next chapter you will learn about how client applications can control and process media themselves by registering DMCC devices in Client Media mode.

Chapter 7

Client-side Media Control

In this chapter:

- Introducing client-side media control
- Exploring the **Client Media** tab
- Getting ready to exercise the DMCC service's client-side media control capabilities
- Viewing RTP and RTCP data
- Redirecting media streams to alternative locations

In the previous chapter, you learned how the DMCC service can be used by client applications to access the media processing resources provided by AE Services. Server-side media control is typically suitable for small applications and for demonstration purposes: if you want to include call recording, media analysis or other media-related functionality in an enterprise-level application, you'll almost certainly need to use client-side media control instead.

For a client application to be able to control the media at an extension, it must register itself as a DMCC device in Client Media mode. Client Media mode indicates that the client application will handle the media streams at the DMCC device. In practice, the application does this by getting Communication Manager to direct the media at the DMCC device to an IP address where separate, external, media processing resources are located that it can control.

In this chapter, you will learn about the DMCC service's client-side media control capabilities, and how they can be demonstrated using the DMCC Dashboard.

About Client-side Media Control

When a call is established, the summed audio media from the other participants in the call is available at each of the extensions involved in the call. Thus, when a client application registers itself as a DMCC device in Client Media mode, it is able to access audio media, in the form of RTP data, for each active call that the extension becomes involved in. How the client application receives and handles the RTP data is determined by the RTP media parameters defined for the DMCC device. The RTP media parameters are initially set when the DMCC device is registered in Client Media mode, but can subsequently be updated.

Setting RTP Media Parameters

When a client application registers a DMCC device in Client Media mode, it sets the RTP media parameters that determine how a call's media is received and where it is sent to by Communication Manager. The RTP media parameters that can be set are:

List of Supported Codecs: A codec is an algorithm used to encode and decode audio media flowing from and to the DMCC device. At registration, the application can specify the list of codecs it supports. Communication Manager maintains separate lists of codecs supported for each type of device in each Network Region. Communication Manager determines the actual codec that is used, selecting a codec that is common to both lists. If there are no common entries, Communication Manager selects a null codec causing no media to be sent or received.

Note: The DMCC Dashboard user interface only allows one supported codec to be selected when registering a DMCC device. However, you can specify a list of supported codecs by submitting a raw XML request.

RTP IP Address and Port Number: Defines the IP address to which Communication Manager will send Real Time Protocol (RTP) data from the DMCC device. RTP is the protocol used to transport media data.

Note: When the DMCC Dashboard is first started, the RTP IP Address defaults to the IP address of the machine on which the DMCC Dashboard is running.

RTCP IP Address and Port Number: Defines the IP address to which Communication Manager will send Real Time Control Protocol (RTCP) data at the DMCC device. RTCP provides control information for an RTP session: it does not transport media data itself. RTCP provides information about the quality of the RTP data.

Note: When the DMCC Dashboard is first started, the RTCP IP Address defaults to the IP address of the machine on which the DMCC Dashboard is running.

Packet Size: Sets the size, in milliseconds, of packets exchanged between the client application and Communication Manager. Valid values are 0, 10, 20, 30, 40, 50 and 60. If 0 or not set, Communication Manager determines the packet size.

Media Encryption: The application can specify whether or not it wants to encrypt the media streams at the DMCC device. If encryption is required, the Advanced Encryption Standard is used.

Note: The DMCC Dashboard user interface does not allow you to set the encryption parameters. Encryption defaults to none. However, you can set encryption by submitting a raw XML request.

Codec G.711U is uncompressed. If you use this codec for your own applications, the audio media is relatively easily converted to PCM (pulse-code modulation) for saving as a WAV file and playing to a speaker. Other codecs are compressed making conversion considerably more complex.

See *Chapter 4: Registering DMCC Devices in Client Media Mode* for instructions on how to register devices, including descriptions of the default RTP media parameter values on the DMCC Dashboard.

Media Events

Client applications that control media themselves need to be informed when the RTP media parameters change at the extensions they are controlling. These parameters change when, for example, a call is started, conferenced, transferred, put on or off hold, and whenever the media path shuffles between direct IP and TDM.

To receive information about RTP parameter changes at an extension, the application must start a Media monitor to listen for `Media Started` and `Media Stopped` events:

- **Media Started Event:** A `Media Started` event is generated whenever a new RTP session is started at the monitored extension. The event is generated not only when the call is first established, but also if the RTP or RTCP media stream is redirected to a different IP address during the call. In the latter case, the RTP session for the old RTP/RTCP IP address is disconnected and a new RTP session started.
- **Media Stopped Event:** A `Media Stopped` event is generated whenever an RTP session is disconnected at the monitored extension. The event is generated when a call is disconnected at the extension and when an RTP media stream is redirected during the call.

Because `Media Started` and `Media Stopped` events do not necessarily indicate that a call has been established or disconnected, they should not be used by the application to trigger media control functionality, such as starting and stopping call recording. Instead, it is recommended that applications start a Call Control monitor to listen for `Established` events to determine when calls start, and to listen for `Connection Cleared` events to determine when calls end.

Media Stream Access Methods

There are two main ways that a client application can access and control the media streams for a call:

- **Multiple Device Registration:** the application registers itself in Client Media mode as a secondary or tertiary DMCC device at an extension. When the extension joins a call, the summed media from all parties in the call is redirected, via the DMCC device, to the RTP IP address specified by the application. This method has the advantage of making efficient use of resources, but does not allow the application to talk. So, for example, the application cannot play a tone or message notifying participants that their call is being recorded.
- **Standalone Device Registration:** the application registers itself as the only device at an extension. This standalone DMCC device may be registered with the Dependency mode set to "Main" or "Independent". When the client application wants to control the media, it adds the standalone DMCC device as a participant in the call. Communication Manager directs the summed media from the other participants in the call to the location specified in its RTP media parameters. This method has the disadvantages of requiring additional extensions to be dedicated to media control and of using

one of the available participant slots in calls (Communication Manager allows up to six participants in a single call). However, the extension can be configured to talk in the calls and thereby allow the application to notify participants when their call is being recorded.

Various methods can be used to determine when a call is established at an extension and to add a DMCC device to a call. There isn't space in this book to go into the details of these methods—they're described on the DevConnect web site (www.avaya.com/devconnect)—but it is important to appreciate the distinction between using multiple and standalone device registration for the purposes of demonstrating client-side media control.

The Client Media Tab

The **Client Media** tab on the DMCC Dashboard comprises the controls used to redirect and view the RTP media streams from a selected Device ID.

The fields on the **Client Media** tab allow you to view RTP and RTCP data from a selected DMCC device that has been directed to the DMCC Dashboard, and to redirect RTP and RTCP data to alternative IP addresses. These options are described in detail later in the chapter.

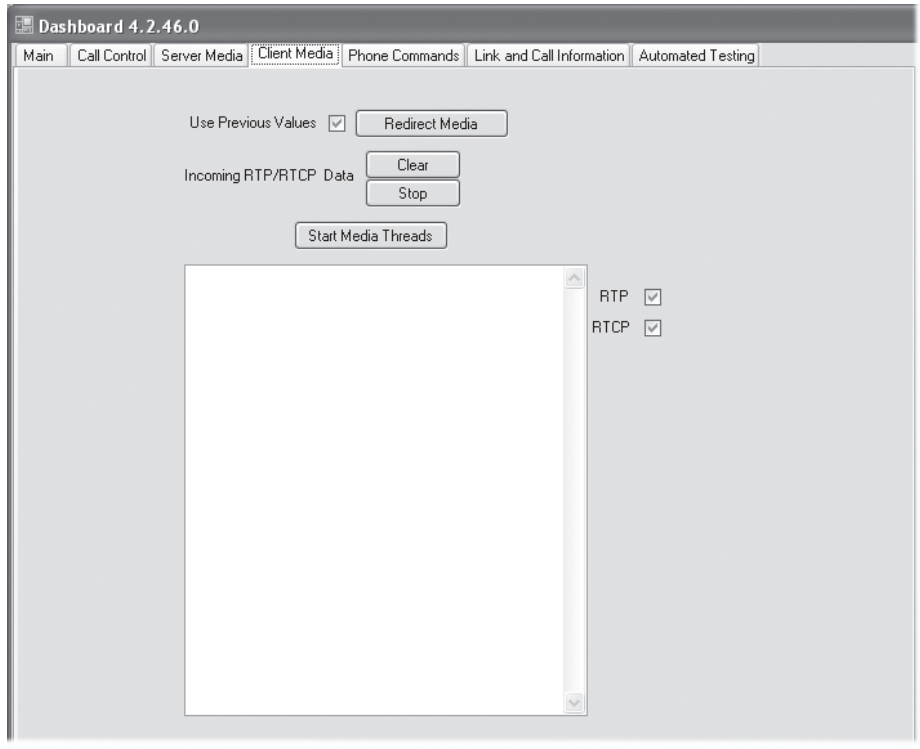

Figure 7-1: The **Client Media** Tab

In future releases of the DMCC Dashboard (post 4.2 release), the **Client Media** tab will include controls that allow you save the media to a WAV file and subsequently play the WAV file using Windows Media Player.

Client-side Media Control Using Avaya IPCoDE

Throughout this book, Avaya IPCoDE provides the installations of Communication Manager and AE Services used for the examples. One of the limitations of Avaya IPCoDE is that it only supports direct media between two stations. Therefore, it is only possible to demonstrate client-side media control by making a call from an IP telephone to an IP extension which has a standalone DMCC device registered against it.

This is not a real-life scenario in the sense that the standalone DMCC device has no talk capability and therefore doesn't generate any RTP data. However, because the media at the other extensions in a call is summed, the RTP data generated at the IP telephone is available at the DMCC device. By default, the media is directed to the RTP IP addresses specified when the standalone DMCC device was registered; but, as you will see later, it can be redirected to an alternative location if required. The examples used to demonstrate client-side media control on the DMCC Dashboard are based on this scenario.

Preliminary Steps

Before you can use the DMCC Dashboard to exercise the DMCC service's client-side media control capabilities, you must perform the following tasks:

1. Start an application session. See *Chapter 4: Starting Application Sessions.*
2. Get the first-party Device ID of the extension whose media you want to control. See *Chapter 4: Getting Device IDs.*
3. Start the Media and other required events monitors at the extension. See *Chapter 4: Starting and Managing Monitors.* In particular, make sure that the DMCC Dashboard listens for `Media Started` and `Media Stopped` events—see *Media Events* above. Setting monitors is not mandatory for client-side media control. However, to understand how client-side media control works it is important to see

the `Media Started` and `Media Stopped` events that are generated. In addition, any media control application you create will need to listen for Media and Call Control events at the extensions it is controlling.

4. Register the DMCC Dashboard as a DMCC device, in Client Media mode, at the extension whose media you want to control. The Dependency mode depends on whether you are using multiple or standalone device registration to access the media streams. Set the RTP and RTCP IP addresses, Codec and Packet Size to the required values. See *Chapter 4: Registering DMCC Devices* and *Media Stream Access Methods* in this chapter.

These preliminary steps are generic and must be performed by any client application that wants to manage media itself. The next section goes through the preliminary steps in more detail, specifically for the example that is going to be used in order to demonstrate client-side media control.

Client-side Media Control Example

To demonstrate client-side media using the DMCC Dashboard and Avaya IPCoDE:

1. Start an application session.
2. Get the Device ID for extension 32131. Extension 32131 is a softphone-enabled extension that is administered as an Avaya H.323 IP softphone on the Avaya IPCoDE installation of Communication Manager. There must not already be a phone set or other device registered as the Main device at the extension.
3. Start the Media, Call Control and Phone monitors at extension 32131.
4. Register the DMCC Dashboard as a DMCC device at extension 32131, in Client Media mode with Dependency mode “Main”. Set the RTP and RTCP IP addresses to the location where the DMCC Dashboard is running. This will cause RTP and RTCP data to be directed to the DMCC Dashboard, allowing it to be viewed at the **Client Media** tab.
5. Start an Avaya IP softphone and log in to extension 32129.
6. Use the Avaya IP softphone to dial extension 32131.
7. Use the DMCC Dashboard IP softphone to answer the call at extension 32131.

A `Media Started` event and an `Established` event are generated at extension 32131, and the details displayed in the **Events** list box. The `Media Started` event details the RTP media parameters that are being used:

```
Media Started Event: 32131:cmsim:192.168.17.129:0
  Monitor ID: 6
  RTP IP Address: 192.168.201.22 RTP IP Port: 4725
  RTCP IP Address: 192.168.201.22 RTCP IP Port:4726
  Packet Size:20 Codec:g711U
    Encryption Information
      Protocol: none
      Payload Type: -1
      Receive Key:
      Transmit Key:
Established Event: 7
     Connection Id: Device Id: 32129:CMSIM::0 Call Id:3838
     Alerting Device Id: 32129:CMSIM::0
     Calling Device Id: 32131:cmsim:192.168.17.129:0
     Called Device Id: 32129:CMSIM::0
     Last Redirection Device Id:
     Event Cause: newCall:
```

Now that the call has been established, you can use the options on the **Client Media** tab to view and redirect the RTP and RTCP data.

Viewing RTP and RTCP Data

To view the RTP and/or RTCP data at the DMCC Device, at the **Client Media** tab:

1. In the **Device IDs** list box, select the first-party Device ID of the extension.
2. Select the **RTP** and/or **RTCP** checkboxes as required to view one or both types of data.
3. Click **Start Media Threads**. This starts the media processing threads in the DMCC Dashboard.
4. Click **Show Incoming RTP/RTCP Data**.

The RTP and/or RTCP data are displayed in the list box at the center of the tab, confirming that the media streams are being directed as expected.

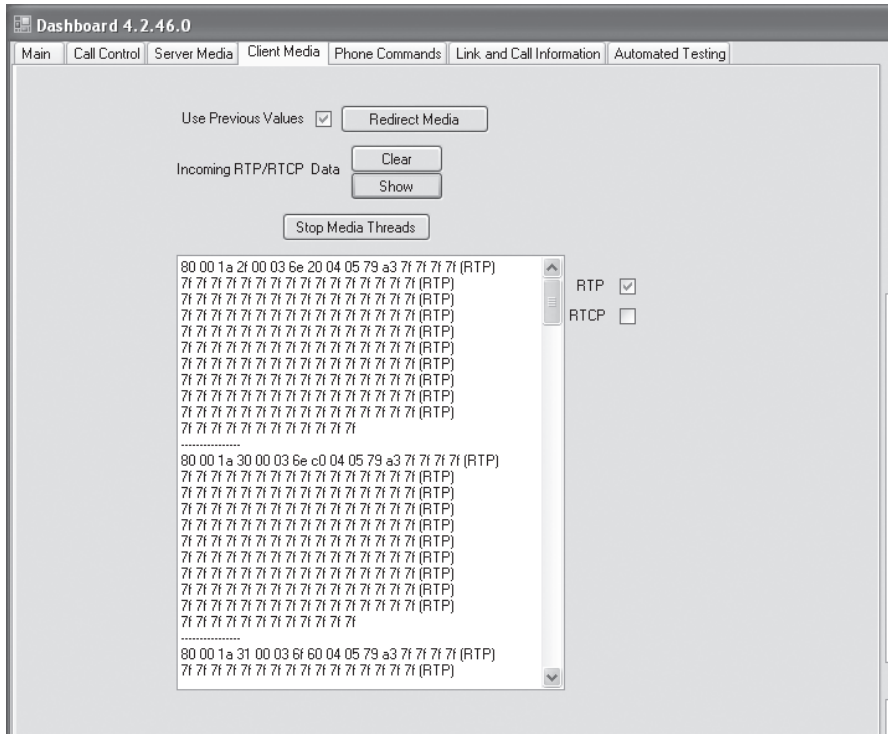

Figure 7-2: Displaying RTP Data

Redirecting Media Streams

The DMCC service also allows you to direct the RTP and RTCP data to alternative locations by disconnecting the current RTP session and starting a new one with different RTP media parameters. To demonstrate redirecting media using the DMCC Dashboard:

1. At the **Main** tab, amend the **Codec, RTP IP Addr.**, **RTP Port, RTCP IP Addr., RTCP Port** and/or **Packet Size** field values as required for the new RTP session.
2. At the **Client Media** tab, select **Use Previous Values** if you want to start an RTP session with the updated RTP and RTCP IP addresses, but the previous codec, packet size and encryption method. Deselect **Use Previous Values** if you want to start a new RTP session using all of the updated RTP media parameters.

3. Click **Redirect Media**.

A `RedirectMediaRequest` message is sent to the server. The current RTP session is disconnected and a `Media Stopped` event generated. A new RTP session is started with the new RTP media parameters and a `Media Started` event generated. Media from the current call is now directed to the new RTP and RTCP IP addresses and any codec or packet sizes changes applied.

Round-up

In this chapter, you learned about the DMCC service's client-side media control capabilities and how to demonstrate them using the DMCC Dashboard. The chapter described how separate media processing streams from calls can be directed to an address where separate media processing resources controlled by the client application are located. In this way, the client application can use the external processing resources to analyze media data, record calls, play messages, collect DTMF tones and perform other media-related tasks.

In the next chapter, you will learn about first-party device control, where a client application can take direct control of the phone set at an extension.

Chapter 8

First-party Device Control

In this chapter:

- Understanding first-party device control
- Exploring the **Phone Commands** tab
- Getting ready to exercise first-party device control capabilities
- Getting information about the state of phone sets and their buttons
- Controlling phone sets by emulating button pushes
- Using the registration and security code options on the **Phone Commands** tab

With first-party device control, a client application takes direct control of a station, including the phone set logged into that station. By controlling the station, the application can monitor the state of the physical elements of a phone set and activate those physical elements to make and manage calls. This chapter describes the DMCC service's first-party device control capabilities and how they can be exercised using the DMCC Dashboard.

The first-party device control options are located in two locations on the DMCC Dashboard: on DMCC Dashboard IP softphone and on the **Phone Commands** tab. The **Phone Commands** tab also includes registration and security code options that are described in this chapter.

About First-party Device Control

First-party device control involves the following activities:

- Getting information about the status of physical elements on phone sets, such as button, lamp and hookswitch states.
- Monitoring phone set events, such as when the phone starts ringing or lamp state changes.
- Activating physical elements of the phone, such as emulating going off hook and pushing buttons.

First-party Device Control in Applications

First-party device control capabilities can be leveraged in a variety of types of client applications:

- **Call recording, using the Service Observing method:** the client application can use first-party device control to activate Service Observing at the observing station. Service Observing is described later.
- **Specialist console or softphone:** the client application has a user interface that allows a user to monitor and control their physical set from their desktop computer. The console may provide enhanced features and services that are not available at the phone set itself.
- **Click-to-dial:** the application could, for example, allow the user to search a directory of employees. When the user clicks on the required employee's details, the application takes control of the phone set, takes it off hook and dials the employee's extension.
- **First-party call control:** because first-party device control does not consume TSAPI licenses, it can be used in preference to third-party call control in client applications that are sensitive to cost. See *Chapter 1: First- and Third-party Call Control* for more information.

Getting Physical Element Information

The DMCC service's device control capabilities allow you to get information about the current status of the physical elements on a phone set, including: button information, button lamp statuses, the hookswitch status, the ringer status, the Message Waiting Indicator status and the contents of the top

line of the display. All of these capabilities can be exercised using the options on the **Phone Commands** tab and are described in this chapter.

Example usage: When a specialist console is being initialized, it can use these options to determine the current state of the phone set it is controlling, including its hookswitch and lamp states, the buttons that are provisioned at the extension and their associated functionality, etc.

Monitoring Phone Events

You have already seen how to start monitors at extensions, enabling the DMCC Dashboard to listen for events you are interested in—see *Chapter 4: Starting and Managing Monitors.* Client applications that use first-party device control are particularly interested in monitoring Phone events:

Example usage: To stay synchronized with the state of the phone set at an extension it is controlling, a specialist console must monitor the Phone events at the extension.

Event	Description
Display Update	The display at the monitored extension has changed.
Hookswitch	The monitored extension has gone on or off hook.
Lamp Mode	The state of one of the button lamps at the monitored extension has changed. For example, the lamp has started flashing, fluttering, shining steadily or has gone off. The lamp mode also indicates the lamp color changes: either red (in-use indicator for call appearance button) or green (feature status).
Ringer Status	The ringer at the monitored extension has started or stopped ringing.

Activating Physical Elements

In *Chapter 4: Using the Dashboard IP Softphone* you saw how the DMCC Dashboard IP softphone can be used to go on or off hook at a phone set and emulate pressing fixed buttons, such as those on the keypad, and **Hold**, **Conference**, **Transfer** and **Drop** buttons. Using these options, it is possible to make calls and perform simple call management tasks. In addition, the DMCC service allows you to press any button provisioned at an extension to exercise its associated functionality.

The **Push Button** option on the **Phone Commands** tab is used to demonstrate how the full set of provisioned buttons on a phone set can be pushed by a DMCC client application.

Example usage: In a click-to-call application, the application takes control of the user's phone set to go off hook dial the extension associated with the selected record.

The Phone Commands Tab

The **Phone Commands** tab on the DMCC Dashboard contains first-party device control, registration and security code options.

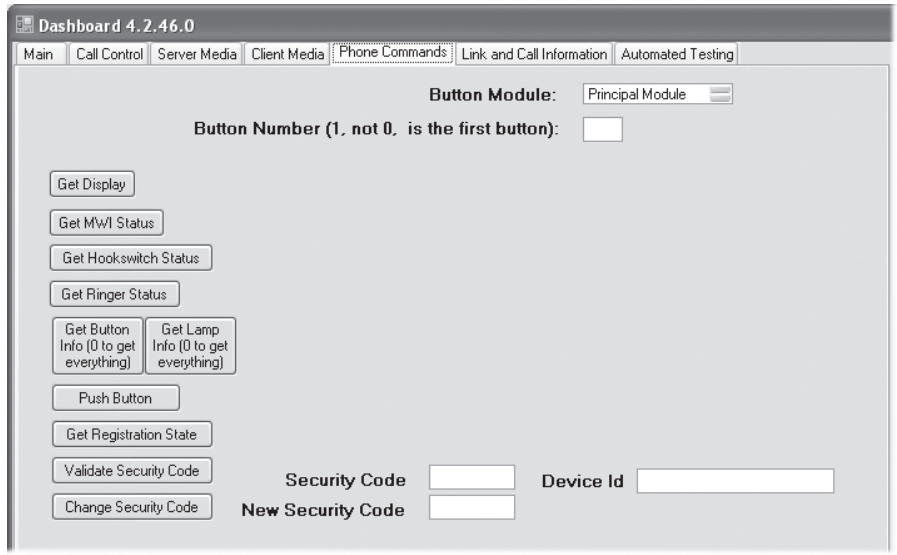

Figure 8-1: The **Phone Commands** Tab

The options at the top of the tab are used to get information about the current state of a phone set, including its buttons and lamps.

The **Push Button** option is used to programmatically push a specified button on a phone set. The **Button Module** and **Button Number** fields are used to specify which button you want to push or get information about.

The tab also contains a registration option that allows you to determine whether the DMCC Dashboard is registered as a DMCC device at an extension; and security code options that allow you to validate or change the security code at an extension.

Each of these options is described in detail later in this chapter.

In future releases of the DMCC Dashboard (post 4.2 release), the **Phone Commands** tab will include fields that allow the user to see information about the state of the phone set at a selected extension. The details will include the contents of the phone display, the ringer state, and information about the first eight buttons and their lamp states.

Identifying Buttons

The **Button Module** and **Button Number** fields at the top of the **Phone Commands** tab are used to specify which button on a phone set you want to get information about or want to "push". When you select the **Get Button Info** or **Push Button** options, the DMCC Dashboard converts the Button Module and Button Number values to a Button Identifier, which is then passed in the request submitted to the server. So how do Button Modules and Button Numbers relate to Button Identifiers?

In your own applications, you can either use and convert Button Modules and Button Numbers, as the DMCC Dashboard does, or use Button IDs directly.

About Button Modules and Button Numbers

Avaya Communication Manager provides a structure whereby the buttons provisioned on phone sets, apart from the keypad buttons, are grouped in modules. Different types of phone sets support different types and numbers of modules. Each module, and each button in each module, is assigned a Button ID constant. The Button Modules that can be controlled via the DMCC service, and their Button IDs, are:

Button Module	Button ID
Principal Module	256
Feature Module	512
Display Module	768
Call Coverage Module	1024

The Principal Module has five fixed buttons with preset functionality and labels that cannot be changed. The fixed buttons are assigned as follows:

Button ID	Assignment
257	Redial button
258	Drop button
259	Conference button
260	Transfer button
261	Hold button

In addition, the Message Waiting Indicator lamp is modeled as a fixed button, with Button ID 262.

The keypad buttons are also assigned fixed Button IDs: 0 to 9 for the corresponding digits, 10 for star, and 11 for pound or hash.

The remaining buttons are assigned functions and labels during station administration on Communication Manager. Each module supports up to 24 administered buttons, although an extension will not necessarily be provisioned with all of them. Within each module, the administered buttons are assigned sequential Button Numbers, starting from one.

Deriving Button Identifiers

The DMCC Dashboard calculates the Button ID of a button by adding its Button Number to the Button ID of the last fixed button in the module.

As mentioned above, the Principal Module has six fixed buttons. The Display, Feature and Call Coverage modules have no fixed buttons. This means that Button Number 1 in each module corresponds to the following Button IDs:

Button Module	Button ID of Button Number 1
Principal Module	263 (=256 + 6 + 1)
Feature Module	513 (=512 + 0 + 1)
Display Module	767 (= 768 + 0 + 1)
Call Coverage Module	1025 (=1024 + 0 + 1)

So, for example, if you ask for information about Button Number 24 in the Principal Module, the DMCC Dashboard submits the request for Button ID 286 (= 256 + 6 + 24).

```
<GetButtonInformation
     xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
     xmlns:xsd=http://www.w3.org/2001/XMLSchema
     xmlns="http://www.ecma-
     international.org/standards/ecma-323/csta/ed3">
       <device typeOfNumber="other" mediaClass="notKnown">
         40010:cmsim:192.168.17.129:0
       </device>
       <button>286</button>
  </GetButtonInformation>
```

Preliminary Steps

Before you can use the DMCC Dashboard to exercise first-party phone control options, you must perform the following tasks at the **Main** tab:

1. Start an application session. See *Chapter 4: Starting Application Sessions.*
2. Get the first-party Device IDs of the extensions you want to control. See *Chapter 4: Getting Device IDs.*
3. Start Phone event and other required monitors at the extensions you want to control. See *Chapter 4: Starting and Managing Monitors.* Setting monitors is not mandatory, but seeing the events generated will help you understand how first-party device control works. Additionally, any applications you create that use first-party device control will need to listen for Phone events.
4. Register the DMCC Dashboard as a DMCC device at each of the extensions you want to control. The devices can be registered in any media mode, although "No Media" is suitable for the examples in this chapter. See *Chapter 4: Registering DMCC Devices.*

Device registration is **not** required if you only want to use the registration or security code options on the **Phone Commands** tab.

The examples in this chapter will use the DMCC Dashboard to control Avaya IPCoDE DCP extension 40010.

Once the preliminary steps have been completed, you can use the options on the DMCC Dashboard IP softphone and on the **Phone Commands** tab to demonstrate the full range of DMCC service's first-party device control capabilities.

Getting Information about Administered Buttons

The options on the **Phone Commands** tab allow you to get information about all of the administered buttons on a phone set or about a single, specified button. Just to remind you, administered buttons differ from fixed buttons in that they are assigned functions and labels during station administration on Communication Manager. You cannot get information about fixed buttons using the options on the **Phone Commands** tab.

To get information about fixed buttons you can submit a raw `GetButtonInformation` XML request directly to the server, passing in the appropriate Button ID. See *Chapter 4: Sending XML Messages Direct to the Server* for information about how to do this.

For each button, the following information is returned:

- **Button:** the Button ID of the button.
- **Button Label:** the label assigned to the button during station administration, if any.
- **Button Label Settable:** indicates whether or not a label can be assigned to the button during station administration.
- **Button Associated Number:** the dial string associated with the button, if any.
- **Button Associated Number Settable:** indicates whether or not a dial string can be associated with the button during station administration.
- **Button Function:** a numerical identifier that indicates which function the button performs when it is pressed, such as call forwarding or sending all calls. The button function may have a constant associated with it that indicates what the functionality is. For example, for the Java API, button function constants are defined in `com.avaya.csta.physical.ButtonFunctionConstants`.
- **Lamp List:** The identifiers of the lamps associated with the button, if any. If there are any associated lamps, their IDs are the same as the Button ID.

Getting Information about a Single Button

To get information about a particular administered button on a phone set:

1. In the **Device IDs** list box, select the ID of the DMCC device that is registered at the phone set's extension.
2. Select the **Button Module** from the list.
3. Enter the **Button Number**. See *Identifying Buttons* above for information about Button Modules and Button Numbers.
4. Click **Get Button Info**.

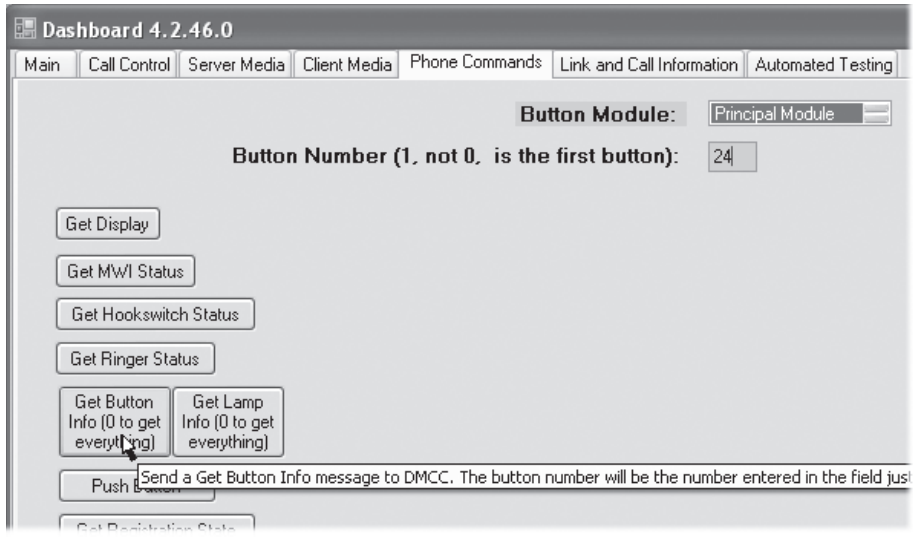

Figure 8-2: Getting Button Information

The DMCC Dashboard converts the Button Module and Button Number values to a Button ID, and sends a `GetButtonInformation` message to the server, passing in the Button ID as parameter. Provided the Button ID relates to an administered button on the phone set, the response message contains information about the button; otherwise, an error message is returned:

```
<GetButtonInformationResponse xmlns="http://www.ecma-
          international.org/standards/ecma-323/csta/ed3">
  <buttonList>
    <buttonItem>
      <button>286</button>
      <buttonLabel></buttonLabel>
      <buttonLabelSettable>false</buttonLabelSettable>
      <buttonFunction>93</buttonFunction>
      <buttonAssociatedNumber typeOfNumber="dialingNumber"
           bitRate="constant">9</buttonAssociatedNumber>
      <buttonAssociatedNumberSettable>
         False
      </buttonAssociatedNumberSettable>
      <lampList>
        <lampID>286</lampID>
      </lampList>
    </buttonItem>
  </buttonList>
</GetButtonInformationResponse>
```

Summary information is displayed in the **Events** text box:

```
Received Button information for 32129:cmsim:192.168.17.129:0
      ID: 286    Label:     Associated Number:    Function: 93
(MANUAL_IN)    lampIds: 286
```

Notice that the summary information includes an abbreviated description of the button functionality. This is inserted by the DMCC Dashboard, which maintains its own list of descriptions for the function identifiers. The description is not available in the response itself.

Example usage: A specialist console application is initialized with, and subsequently monitors, the state of a phone set it is controlling. At some point in the future, a new button is provisioned at the phone set's station, but the application is not notified. The first time the button is used, an associated `Lamp Mode` event is generated. This triggers the application to get information about the button associated with the lamp and update its view of the phone set.

Getting Information about all Buttons

To get information about all of the administered buttons that have been provisioned on a phone set:

1. In the **Device IDs** list box at the top right-hand side of the DMCC Dashboard user interface, select the ID of the DMCC device that is registered at the phone set's extension.
2. Enter "0" in the **Button Number** field, or leave the field blank. The value in the **Button Module** field is ignored—information is returned for all administered buttons in all modules.
3. Click **Get Button Info**.

A `GetButtonInformation` message is sent to the server. The response message contains detailed information about all of the administered buttons and the DMCC Dashboard displays a summary in the **Events** text box:

```
Received Button information for  32129:cmsim:192.168.17.129:0
    ID: 1026    Label:     Associated Number:
         Function: 15 (NORMAL)        lampIds: 1026
    ID: 1025    Label:     Associated Number:
          Function: 66 (LAST_NUMB)     lampIds: 1025
    ID: 265     Label:     Associated Number: 32129
          Function: 6  (CALL_APPR)     lampIds: 265
    ID: 264     Label:     Associated Number: 32129
          Function: 6  (CALL_APPR)     lampIds: 264
    ID: 263     Label:     Associated Number: 32129
          Function: 6  (CALL_APPR)     lampIds: 263
    ID: 262     Label:     Associated Number:
          Function: 5  (MWI)           lampIds: 262
```

Getting Information about Lamp States

The DMCC service allows you to get information about the state of the lamps associated with a particular administered button or with all administered buttons on a phone set. For each lamp, the following information is returned:

- **Lamp:** the Lamp ID of the lamp.
- **Lamp Mode:** indicates how the lamp is lit. For example, `Steady`, `Off`, `Flutter`, etc.
- **Lamp Color:** the color of the lamp, either red or green. A separate lamp status record is returned for each color.
- **Button:** the Button ID of the button the lamp is associated with.

The methods for getting the states of the lamps associated with one or all buttons are similar to those for getting information about the buttons themselves, except that you click **Get Lamp Info** instead of **Get Button Info**.

Getting Other Information about Phone Sets

As well as button and lamp status information, the DMCC service allows you to get other information about the current state of a phone set. The following options are available on the **Phone Commands** tab to demonstrate these capabilities:

- **Get Display:** returns a snapshot of the contents of the top line of the phone set display.
- **Get MWI Status:** returns the status of the phone set's Message Waiting Status lamp: either `True` (the lamp is on because there are outstanding voicemail messages) or `False` (the lamp is off because there are no outstanding voicemail messages).
- **Get Hookswitch Status:** returns the state of the phone set's hookswitch: either `True` (on hook) or `False` (off hook).
- **Get Ringer Status:** returns the current status of the ringer associated with the phone set, including the ring mode (`Ringing` or `Not Ringing`) and the ring pattern. The ring pattern is identified by a numerical constant, for example, "0" (ringer off), "11" (standard ring), "13" (priority ring), etc.

To use any of these options, select the Device ID associated with the phone set and click the appropriate button on the **Phone Commands** tab. A corresponding message is sent to the server (`GetDisplay`, `GetMessageWaitingIndicator`, `GetHookswitchStatus`, or `GetRingerStatus`). The DMCC Dashboard extracts the information from the response message and displays it in the **Events** text box.

Controlling Phone Sets

WWhen using first-party device control, a DMCC client application can take direct control of the phone set at an extension, including programmatically pushing the buttons on the phone set. In this way, the application can control all of the features of the phone set, including the ability to make and manage calls.

In *Chapter 4* you saw how the DMCC Dashboard IP Softphone can be used to emulate pushing keypad buttons and the fixed **Transfer**, **Conference**, **Hold** and **Drop** buttons. The DMCC Dashboard IP softphone can also be used to take the handset off and on hook at a speakerphone-equipped extension. See *Chapter 4: Using the DMCC Dashboard IP Softphone* for more information.

This section describes how the DMCC Dashboard can be used to emulate pushing administered buttons on a phone set. Using the DMCC Dashboard IP softphone and **Push Button** option on the **Phone Commands** tab, you can exercise the full range of phone set control options.

Pushing Administered Buttons

To emulate pushing one of the administered buttons on a phone set:

1. In the **Device IDs** list box, select the ID of the DMCC device that is registered at the phone set's extension.
2. Select the appropriate **Button Module** from the list.
3. Enter the **Button Number**. See *Identifying Buttons* above for information about Button Modules and Button Numbers.
4. Click **Push Button**.

The DMCC Dashboard converts the Button Module and Button Number values to a Button ID, and sends a `ButtonPress` message to the server, passing in the Button ID as parameter:

```
<ButtonPress xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xmlns:xsd=http://www.w3.org/2001/XMLSchema
  xmlns="http://www.ecma-international.org/standards/ecma-
  323/csta/ed3">
    <device typeOfNumber="other" mediaClass="notKnown">
      32129:cmsim:192.168.17.129:0
    </device>
    <button>286</button>
</ButtonPress>
```

Provided the Button ID relates to an administered button on the phone set, the extension behaves as though the button had been physically pushed on the phone set.

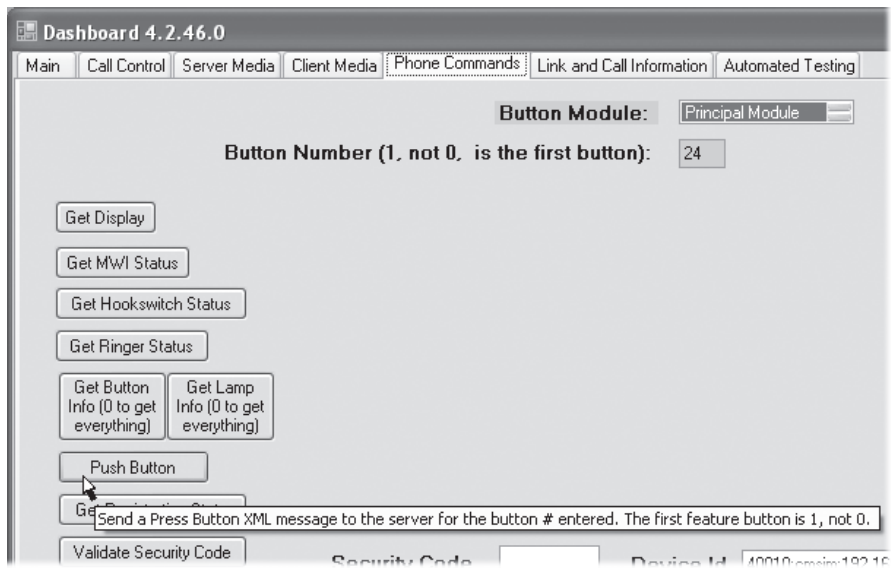

Figure 8-3: Pushing an Administered Button

Which Button to Press?

As discussed earlier, functionality is assigned to administered buttons during station administration on Communication Manager. Therefore, a client application cannot know what a particular button does just from its Button ID; it has to determine for itself which button it needs to press to invoke a particular function. Let's use the example of activating Service Observing to explain how a client application might do this.

Service Observing is primarily used to allow an external party to listen in on calls at a specified extension. To support this feature, the observing extension can be provisioned with a **Service Observing** button. Service Observing may be activated in different ways, but in this example it is activated by pressing the **Service Observing** button at the observing extension and dialing the number of the extension to be observed. Whenever the observed extension joins a call, the observing extension is automatically added and is able to listen in.

A client application can activate Service Observing, using first-party device control to press the **Service Observing** and keypad buttons. But how does the application know which button is the **Service Observing** button and therefore the correct one to press? One way the client application could determine and press the correct button is as follows:

1. Use the **Get Button Information** option to get information about all of the buttons at the observing extension.
2. Iterate through the results to locate the button that implements Service Observing functionality. In this case, it looks for the button that has been assigned function identifier "85", which always corresponds to Service Observing.
3. Get the matching button's Button ID.
4. Use the **Button Press** option, passing in the Button ID.

The application then presses appropriate keypad buttons to dial the observed extension number and activate Service Observing.

Registration and Security Code Options

In addition to third-party control options, the **Phone Commands** tab also includes a number registration and security code options, namely:

- **Get Registration State**
- **Validate Security Code**
- **Change Security Code**

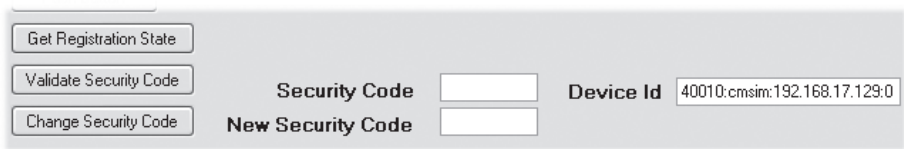

Figure 8-4: Registration and Security Code Options

Getting the Registration State

The **Get Registration State** option is used to determine whether or not a DMCC device is registered at the selected extension. To use this option you must first select the extension's Device ID in the **Device IDs** list box. A `GetRegistrationState` message is sent to the server which returns a registration state of `IDLE`, `REGISTERED`, `REGISTERING` or `UNREGISTERING`.

Example usage: DMCC client applications may need to check whether or not they are registered as a DMCC device before attempting to perform certain tasks.

Validating and Changing Security Codes

Extensions may be provisioned with security codes during station administration on Communication Manager. To use an extension that has been provisioned with a security code, the user must enter the appropriate password. Security codes are used to prevent unauthorized access to the system.

If a security code has been provisioned at an extension, you can demonstrate how a DMCC client application can validate or change it using these options:

- **Validate Security Code:** check that the password entered in the **Security Code** field is correct for the extension identified by its **Device ID**.
- **Change Security Code:** change the password at the extension identified by its **Device ID** to the password entered in the **New Security Code** field. The old password must be entered in the **Security Code** field.

Round-up

In this chapter you were introduced to the DMCC service's first-party device control capabilities and learned how to exercise them on the DMCC Dashboard.

The next chapter looks at how information about active calls can be retrieved by DMCC client applications.

Chapter 9

Getting Active Call Information

In this chapter:

- Understanding call information links
- Exploring the **Link and Call Information** tab
- Getting link status information.
- Getting call information

This chapter covers the final set of AE Services DMCC service capabilities that can be exercised using the DMCC Dashboard. The last two options are grouped on the **Link and Call Information** tab. These options allow you to get information about the status of the call information link between Communication Manager and the AE Services server, and to get information about active calls at selected extensions.

About Call Information Links

Whenever you start a new application session, various communication links are automatically established between the AE Services server and Communication Manager. One of these links is the call information link. The call information link allows DMCC client applications to get information about active calls.

The DMCC service allows a client application to check whether or not the call information link is operational and, therefore, whether or not it can get information about active calls.

A client application could start multiple application sessions, each connected to a different Communication Manager installation. Therefore, there may be multiple call information links for each switch connection. The DMCC service allows client applications to check the status of all call information links or just a specific link.

In addition, a client application can start a Call Information monitor to listen for `Link Down` and `Link Up` events and thereby track the status of the call information link—See *Chapter 4: Starting and Managing Event Monitors.* A `Link Down` event is generated whenever the call information link goes down, and indicates which link failed and whether or not the server will attempt to reconnect the link automatically. If the link is reestablished, a `Link Up` event is generated.

The DMCC service uses the TSAPI service, also on the AE Services server, to provide its third-party call control, logical device and snapshot capabilities—see *Chapter 5: Third-party Call Control.* These capabilities require separate links (known as "t-links") to be operational between the AE Services server and Communication Manager. The DMCC service does not provide an option to check or monitor the status of t-links. However, call information links and t-links are on the same transport layer so it is likely, although not guaranteed, that they will have the same status.

The Link and Call Information Tab

The **Link and Call Information** tab on the DMCC Dashboard contains Call Information and Link Status options.

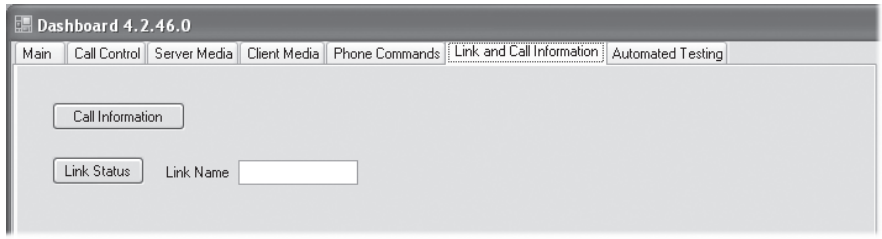

Figure 9-1: The **Link and Call Information** Tab

Each of these options is described in detail in the sections that follow.

Preliminary Steps

Before you can use the DMCC Dashboard to get information about the active call at an extension, you must perform the following tasks at the **Main** tab:

1. Start an application session. See *Chapter 4: Starting Application Sessions.*
2. Get the first-party Device ID of the extension at which you want to get active call information. See *Chapter 4: Getting Device IDs.*
3. Start the Call Information events monitor at the extension. See *Chapter 4: Starting and Managing Monitors.*
4. Register the DMCC Dashboard as a DMCC device against the extension. See *Chapter 4: Registering DMCC Devices.*
5. Establish an active call at the extension. It doesn't matter how you do this: you could use the DMCC Dashboard's first- or third party call control options or place the call manually.

The only required preliminary step for getting the status of a call information link is to start an application session.

Getting Call Information

There are various reasons why a client application may want to retrieve information about active calls. For example, an application could obtain information about a call that is being recorded and save it with the recording for search and retrieval purposes. Alternatively, a client application could display summary information about all active calls in the call center.

Provided the call information link is operational, you can get information about the active call at an extension:

1. In the **Device IDs** list select the first-party Device ID associated with the extension.
2. On the **Link and Call Information** tab, click **Get Call Information**.

A `GetCallInformation` message is sent to the server. The response message contains information about the active calls at the extension, if any. Extract details from the response message are displayed in the **Events** text box:

```
Call Information Response: 172 agentLoginId:
agentStationExtension: CallId:1183263 Incoming:
False IncomingTrunkCall:False recordingExtension:40010
recordingExtensionName:Station 40010 ServiceObservingActive:
False StationName: UniversalCallId:2814483819660093011
vdnExtension: vdnName:
```

The DMCC service provides an alternate method for getting information about the calls at an extension, using the **Snapshot Device** option—see *Chapter 5: Third-party Call Control.* The **Snapshot Device** option returns information about all calls at an extension, not just the active call, but provides less information about each call. See the API documentation; or use the DMCC Dashboard to try out both methods, to determine which method best meets the requirements of your applications.

Getting Link Status Information

The DMCC service allows you to check the status of the call information links between itself and the Communication Manager installations it is connected to.

On the **Link and Call Information** tab:

1. To get the status of a particular link, enter the switch connection name in the **Link Name** field. The examples in this book use Avaya IPCoDE to provide the switch, so the appropriate link name is `cmsim`. To get information about the status of all links, leave the **Link Name** field blank.
2. Click **Link Status**.

A `GetLinkStatus` message is sent to the server. The response contains the following information about each link:

- **Link Name:** the name of the link.
- **Link Up:** either `True` (up) or `False` (down).
- **Auto Reconnect:** `True` if auto reconnect is enabled; otherwise `False`.

To be able to get information about the calls at an extension, the call information link must be up.

Round-up

In this chapter you were introduced to the DMCC service's call information capabilities and learned how these can be demonstrated on the DMCC Dashboard. You can use these capabilities in your own applications to get information about active calls.

In the next chapter, you will learn how the DMCC Dashboard can help you prototype and test your own DMCC client applications.

Chapter 10

Automated Testing and Prototyping

In this chapter:

- Introducing the DMCC Dashboard's automated testing and prototyping features
- Exploring the **Automated Testing** tab
- Getting XML message information
- Rerunning sequences of actions
- Saving, loading and editing test scripts

This chapter looks at the features of the DMCC Dashboard that are designed to help developers prototype and test their own DMCC client applications. Testing and prototyping options are grouped on the **Automated Testing** tab.

About Automated Testing and Prototyping

During an application session, the DMCC Dashboard captures information about every button pushed on the DMCC Dashboard user interface that results in a DMCC API request being sent to the server, including the context and associated field values. So, for example, if you select the Device ID for extension 40010 and click **Off Hook** on the DMCC Dashboard IP softphone, the DMCC Dashboard captures not only the **Off Hook** button push itself, but also the fact that it resulted in a `SetHookSwitchStatus` command being sent for extension 40010.

To facilitate automated testing, the DMCC Dashboard allows you to rerun sequences of actions that were originally initiated by pushing buttons on the user interface. In addition, you can save a sequence of actions to an XML script that can be loaded and rerun at a later time. If desired, developers can update scripts manually to give them a finer degree of control over their testing.

The DMCC Dashboard also provides advanced options that allow you to rerun a sequence of actions against a different first-party Device ID, edit scripts, run scripts that were generated externally, annotate actions and even control the time interval between actions.

In addition to testing, developers can use the options on the **Automated Testing** tab to prototype features of the applications they are developing. For example, once you have identified the set of actions required to perform a particular task, you can save them to a script. You can then demonstrate your ideas to other interested parties by simply rerunning the script, without the need to write any code.

The Automated Testing Tab

The **Automated Testing** tab displays information that allows you to save and replay series of button pushes to facilitate automated testing.

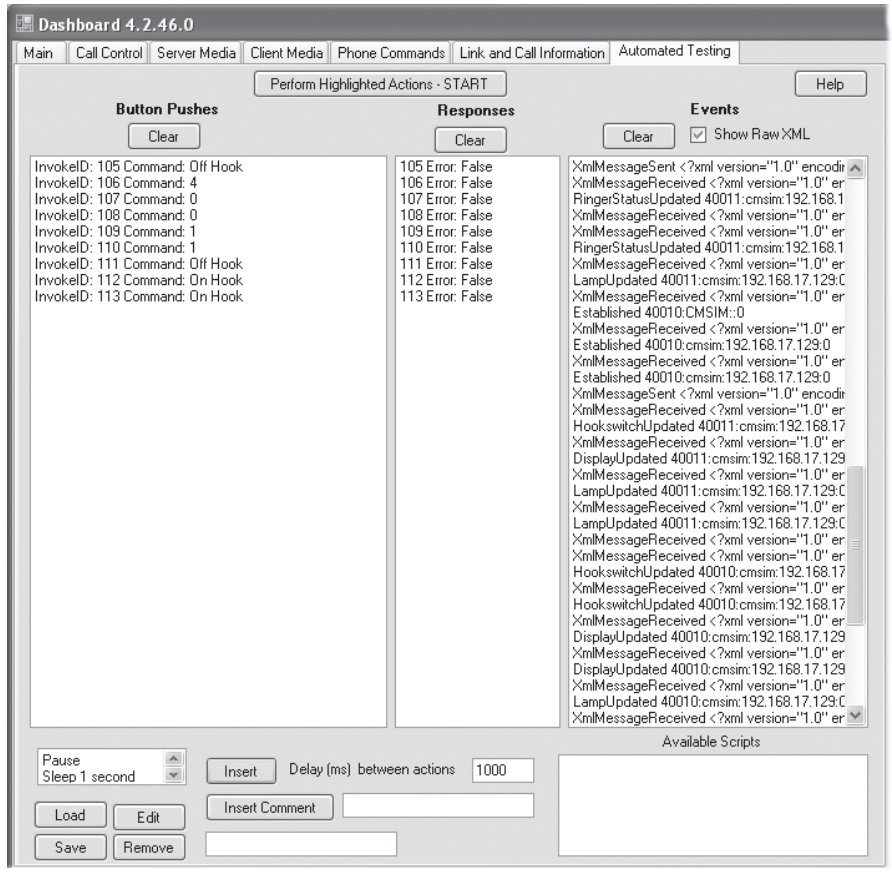

Figure10-1: The **Automated Testing** Tab

The DMCC Dashboard automatically captures the actions associated with button pushes and displays the details in the three main list boxes on the **Automated Testing** tab:

- **Button Pushes:** lists the commands that are invoked for the button pushes. Each command is assigned an Invoke ID, which can be used to match it to the corresponding **Responses** entry. Invoke IDs are sequential numbers, starting from 1 when a new application session is started.
- **Responses:** indicates whether the command with the same Invoke ID was successful (`Error: False`) or failed with an error (`Error: True`). If a command fails, you can view the error details by double clicking on the appropriate entry, as shown in **Figure 10-2.**
- **Events:** if the **Show Raw XML** checkbox is selected, the DMCC Dashboard displays the incoming and outgoing XML messages generated by each button push. If the checkbox is not selected, the DMCC Dashboard displays descriptions of the monitored events generated by each button press.

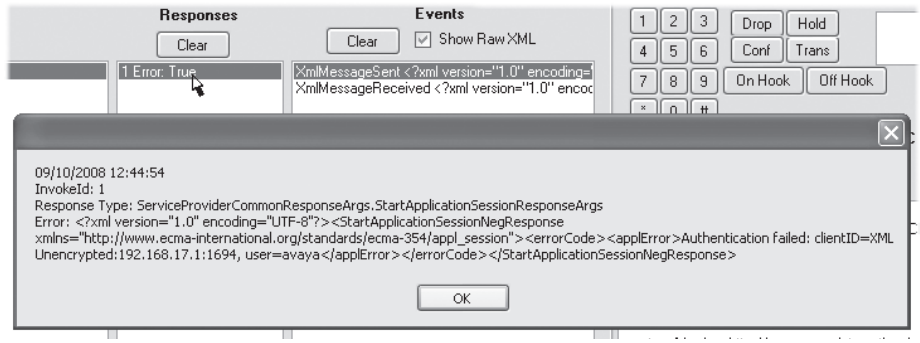

Figure 10-2: Viewing Command Error Details

The **Automated Testing** tab also includes options that allow you to:

- See which XML messages were sent and received in the period between selected button pushes.
- Rerun a sequence of actions
- Insert pauses and comments into a sequence of actions
- Save a sequence of actions to an XML script
- Edit a saved script in a text editor
- Load and run a previously saved script

Each of these options is described in detail in the sections that follow.

The **Help** option on the tab displays automated testing help text in the Exceptions and Errors region of the DMCC Dashboard.

Automated Testing Example

A simple example task will be used to help explain the DMCC Dashboard's automated testing features. The sample task involves establishing and ending a call using first-party call control. The example assumes that event monitors have been started at extensions 40010 and 40011, and that the DMCC Dashboard has registered itself as a DMCC device at both extensions. To make a call from extension 40010 to extension 40011, and thereby generate the actions for the example task:

1. At the **Automated Testing** tab, click **Clear** at the top of the **Button Pushes, Responses** and **Events** list boxes. This clears details of all the actions that have been performed since the application session was started, making it easier to see what is happening for the example.
2. In the **Device IDs** list box on the DMCC Dashboard IP softphone, select the Device ID for extension 40010.
3. Click **Off Hook.**
4. Click keypad buttons **4, 0, 0, 1, 1** to make the call
5. Select the Device ID for extension 40011 and click **Off Hook** to answer the call.
6. Select the Device ID for extension 40010 and click **On Hook** to end the call.
7. Select the Device ID for extension 40011 and click **On Hook** to hang up.

The button pushes are captured and resultant messages displayed on the **Automated Testing** tab, as shown in **Figure 10-3**. As you can see, all of the commands were successful (`Error: False`).

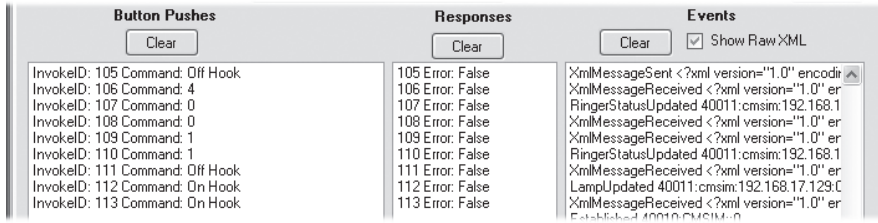

Figure10-3: The Example Button Pushes

Getting XML Message Information

To help understand how an application behaves, it is sometimes useful to see which XML messages were sent and received during a specified period. To see the XML messages that were exchanged between the DMCC Service and the DMCC Dashboard in the period between two button pushes:

1. In the **Button Pushes** list box, click on the first button push to select it.
2. Press **Shift** and click on the second button push to select it.

The XML messages that were sent and received during the period between the first and second button push are highlighted in the **Events** list box.

In addition, to see when a particular message was sent or received, double-click on the message in the **Events** list box. The full content of the XML message and its timestamp are displayed in a message box.

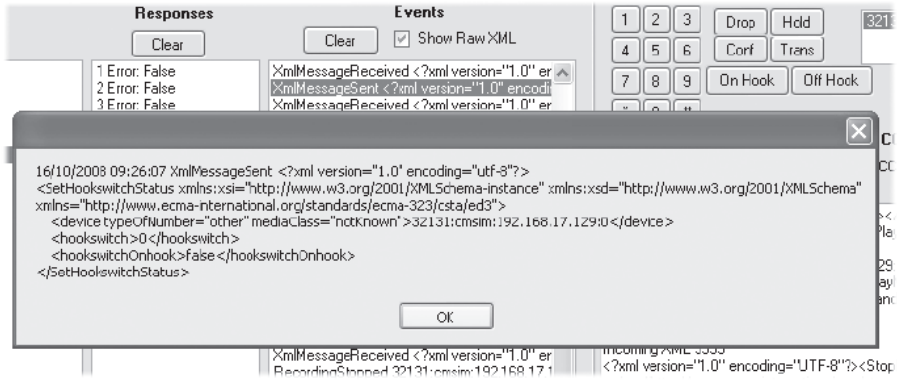

Figure 10-4: Viewing XML Message Details

Rerunning Sequences of Actions

The DMCC Dashboard allows you to rerun a sequence of recently performed actions by emulating the corresponding sequence of button pushes in their original context. To rerun the example task:

1. Highlight the sequence of actions in the **Button Pushes** list box. Notice that the XML messages that were sent and received during the period between the first and last selected button pushes are automatically highlighted in the **Events** list box.

2. Ensure that the DMCC Dashboard and Device IDs are in the correct start state for the task. For the example, this involves ensuring a DMCC device is still registered at both extensions and that both extensions are on hook.
3. Click **Perform Highlighted Actions—START**.

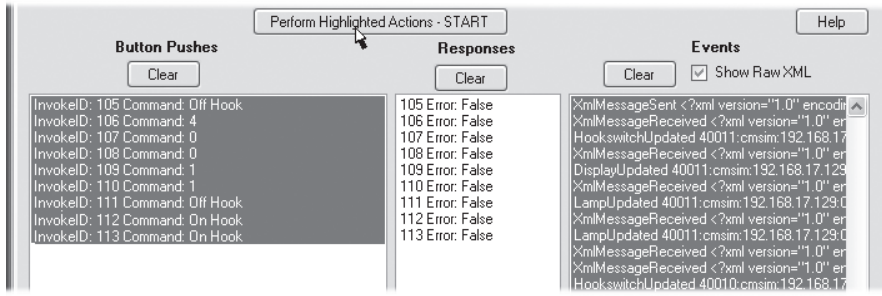

Figure 10-5: Rerunning a Series of Commands

As each action is performed, the corresponding button is highlighted on the Dashboard. **Figure 10-6** shows the IP softphone when the DMCC Dashboard is emulating pushing keypad button 0 at extension 40010.

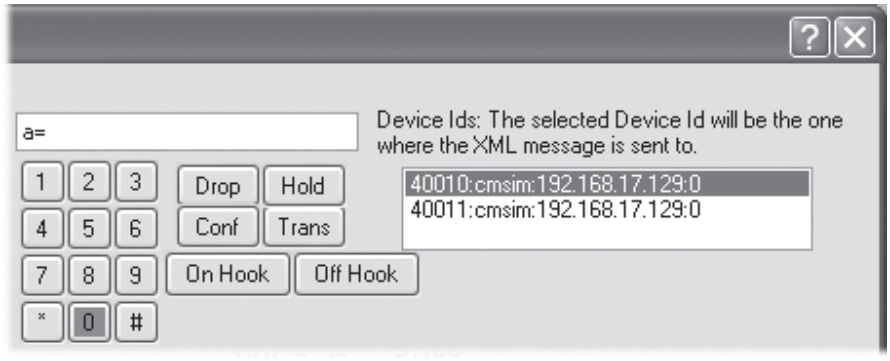

Figure 10-6: Automated Button Presses

When you close the DMCC Dashboard, or clear the **Button Pushes** entries, the actions are no longer available to be rerun. However, you can save actions to a script file for subsequent reloading into the DMCC Dashboard. See *Saving, Editing and Loading Scripts* below.

Rerunning Actions on Different Device IDs

The DMCC Dashboard allows you to see which Device IDs selected actions were originally, or most recently, performed on. You can also change the Device IDs that will be used when the actions are next run. Changing Device IDs allows you to test the same task on different extension types and configurations.

To see which Device IDs that a sequence of actions was originally performed on:

1. Highlight the sequence of actions in the **Button Pushes** list box.
2. Right click on the highlighted actions and select **View Device IDs that will be used**.

A dialog box opens listing the Device IDs, as shown in **Figure 10-7**.

To change the Device ID that one or more actions will use when they are next run:

1. In the **Device IDs** list box, select the new Device ID.
2. In the **Button Pushes** list box, highlight the actions you want to change.
3. Right click on the highlighted actions and select **Change the Device ID that will be used to the currently selected Device ID.**

The next time you run the selected actions, they will use the new Device ID.

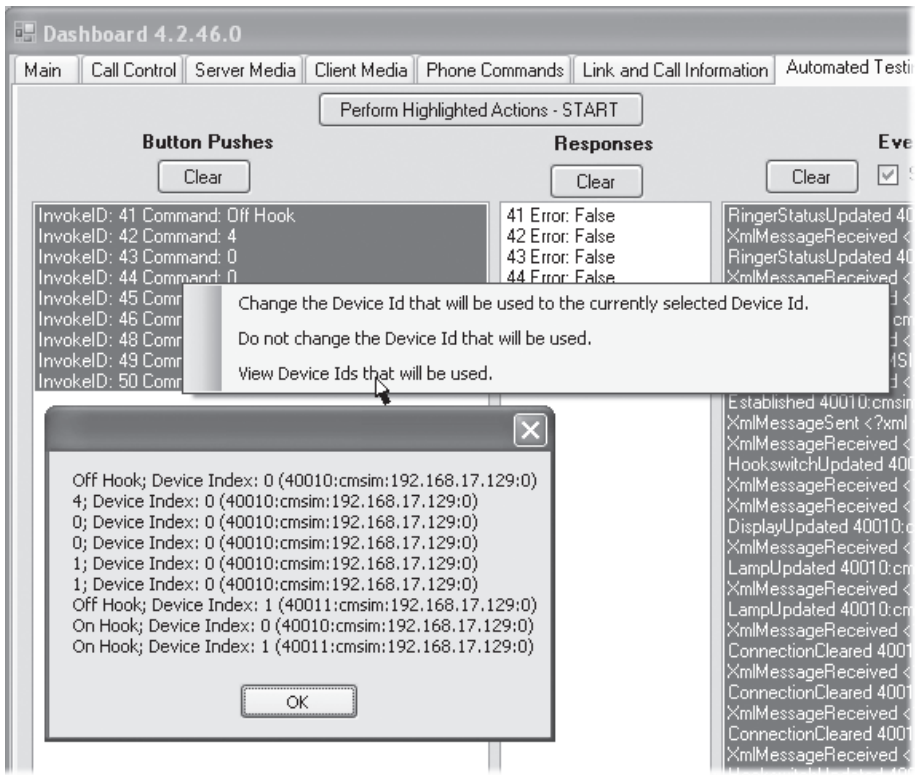

Figure 10-7: Viewing Action Device IDs

Controlling Action Timings

By default, there is a 1000 millisecond (1 second) delay between actions when rerunning a task. The DMCC Dashboard lets you change the default delay and also insert additional pauses between selected actions. This may be desirable if, for example, you need to allow the server more time to respond to commands or wish to slow a task down so that you can observe it in detail.

To change the default delay between actions, simply amend the value in the **Delay between actions** field.

To insert an additional delay between two actions:

1. Highlight the second of the actions in the **Button Presses** list box.
2. Select the required delay period from the list of options located to the left of the **Insert** button: either Pause, 1, 5, 10 or 20 seconds.
3. Click **Insert**.

The delay is inserted before the selected action. **Figure 10-8** shows a 20 second delay inserted between the call being answered at extension 40011 and the call being ended by extension 40010 going on hook.

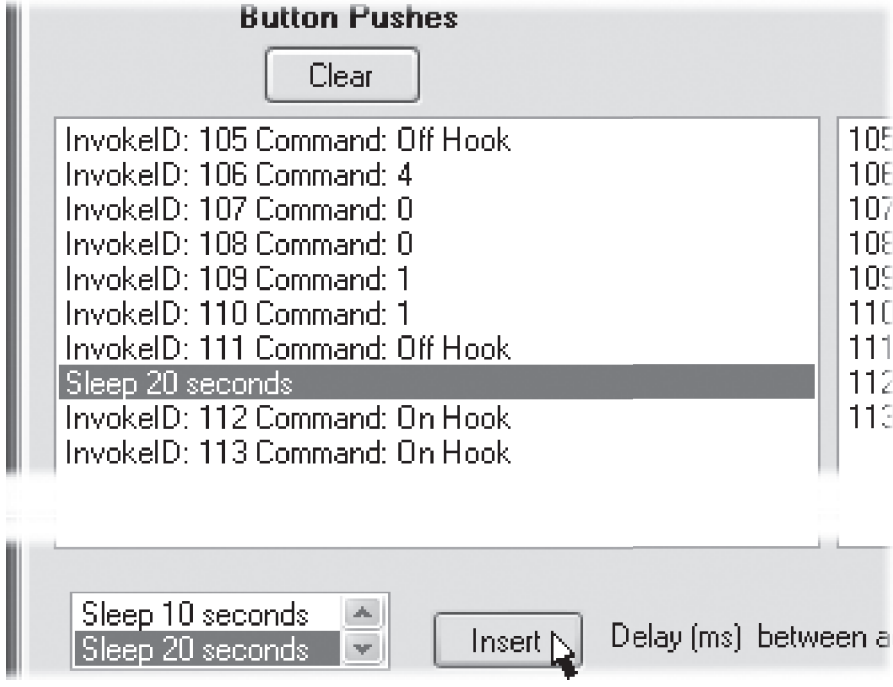

Figure 10-8: Inserting a Delay Between Actions

Note: When you save a sequence of actions to a script, the delays are also saved.

Inserting Comments

The DMCC Dashboard allows you to insert comments into a sequence of actions. When you save a sequence to a script, the comments are also saved. Comments help users understand what is happening, or is supposed to happen, at various points in a task.

To insert a comment before an action:

1. Highlight the action in the **Button Presses** list box.
2. Enter the comment text in the field next to the **Insert Comment** field.
3. Click **Insert Comment**.

Saving, Editing and Loading Scripts

The DMCC Dashboard lets you save sequences of actions to an XML script file. Subsequently, you can load the file and rerun the actions you have saved. You can also edit scripts before you rerun them, to give you finer control over your testing activities, or even create and run your own scripts.

The XML in the script file is not the same as the XML exchanged in messages between DMCC clients and the server. The XML generated when you save a sequence of actions cannot, therefore, be used directly in your applications.

Saving Sequences of Actions

To save a sequence of actions to a script file:

1. Highlight the sequence of actions in the Button Pushes list box.
2. Enter a name for the script in the text box to the right of the **Remove** button. For our example, we'll call the script `MakeCall2`.
3. Click **Save**.

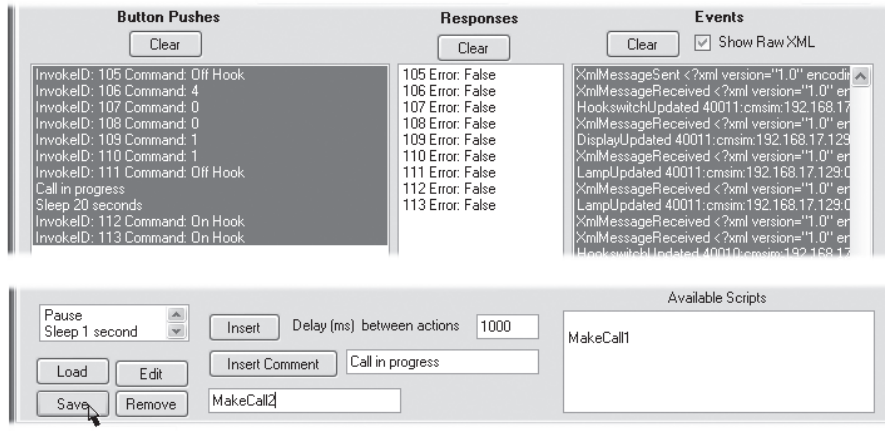

Figure 10-9: Saving a Sequence of Actions to a Script

The sequence of actions is saved in the script file called `MakeCall2.dashboard`, located in the DMCC Dashboard installation directory. The new script is added to the list of **Available Scripts.**

As well as saving your own scripts, you can use scripts from other sources, such as other developers, or even compose them from scratch in a text editor. To make other scripts available, make sure they have been saved with a `.dashboard` extension and copy them to your DMCC Dashboard installation directory.

Editing Scripts

To edit an existing script:

1. Select the script in the **Available Scripts** list box.
2. Click **Edit**.

The script file opens in a text editor.

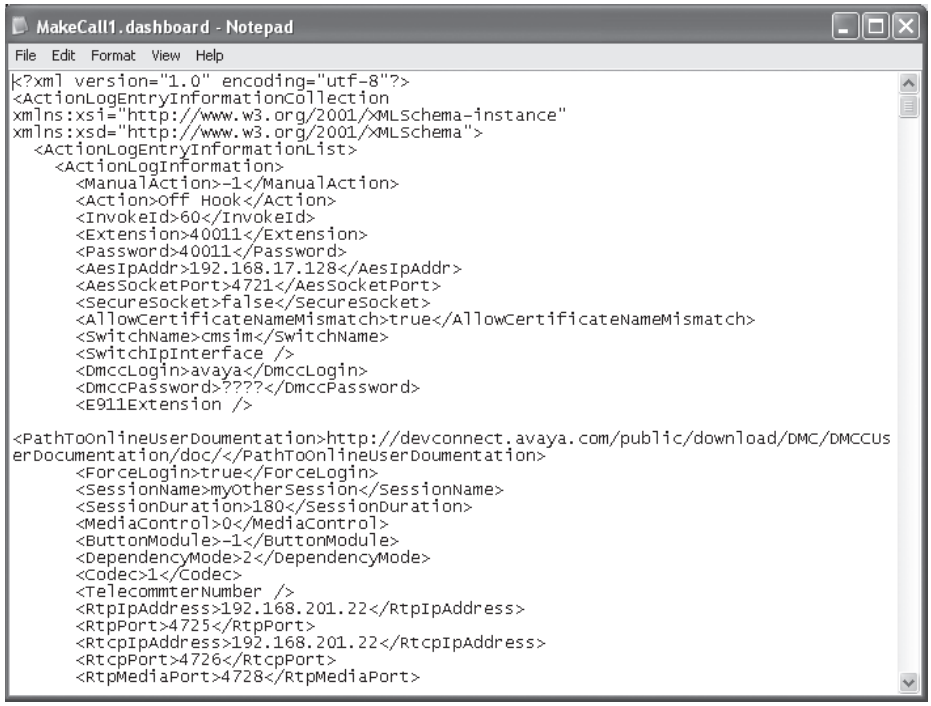

Figure 10-10: Editing a Script File

Although the format of the script file is particular to the DMCC Dashboard, it is relatively easy to interpret. Editing the script file gives you a fine degree of control over how individual actions and sequences of actions are performed. When you save your changes and rerun the script, the DMCC Dashboard allows you to monitor the effects of the changes you have made.

Loading and Running Scripts

To load a script into the DMCC Dashboard:

1. Select the script in the **Available Scripts** list box.
2. Click **Load.**

The actions saved in the script are retrieved and listed in the **Button Pushes** list box.

To run a sequence of actions from the loaded script:

3. Highlight the sequence of actions in the **Button Pushes** list box.
4. Click P**erform Highlighted Actions—START**.

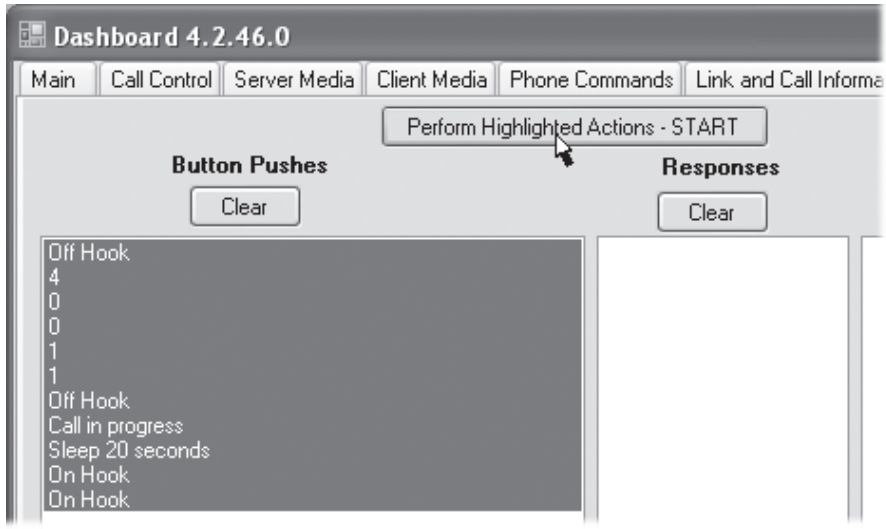

Figure 10-11: Running a Script

As you can see, this option allows you to repeat your automated tests as many times as you wish.

Round-up

In this chapter you were introduced to the DMCC Dashboard's automated testing features and learned how you can use them to help develop your own client applications.

This chapter concludes our exploration of the DMCC Dashboard and how it can be used to help developers create applications that leverage the device, media and third-party call control capabilities provided by the DMCC service.

Chapter 11

Get the Dashboard and Get Started!

Getting Started on Your Own

This concludes our exploration of device, media and call control using the Avaya DMCC Dashboard. By now, you should be comfortable with the concepts of device, media and call control, and ready to use the DMCC Dashboard to increase your mastery of advanced IP communications application development using the Avaya AE Services DMCC service.

So, how do you go about applying what you have learned and start creating exciting applications that deliver intelligent communications capabilities that transform business operations?

Start by joining the Avaya DevConnect community. Being a DevConnect member gives you access to the tools and resources you need to help you create intelligent communications solutions. The Appendix at the end of this book tells you all about DevConnect, the benefits it offers and how to join.

Once you are logged in as a DevConnect member, you can download a copy of the DMCC Dashboard from the DevConnect portal. Use the Products & SDKs Index, under Application Enablement (AE) Services, to find information on the current version available for download. And if you don't have access to an Avaya infrastructure, consider obtaining a copy of the Avaya IP Communications Development Environment (Avaya IPCoDE) so you can fully exercise the dashboard. Again, the Appendix has more information on Avaya IPCoDE and AE Services.

Before we close, however, we'd like to provide you with a recap of some of the essential takeaways, and offer some advice on how to apply what you have learned in your own intelligent communications development projects.

Key Take-aways and Top Ten DMCC Dashboard Facts

There are a couple of key points you should take away from this book:

- AE Services exposes and abstracts the communications capabilities of Avaya Communication Manager, making it possible for you to create applications that seamlessly integrate communications capabilities with your business processes. The AE Services DMCC APIs support a huge range of communications functionality, from basic call control, though call recording and contact center services, to presence-based routing. The DMCC service provides the ideal interface for the creation of applications that enable "intelligent communications."
- The DMCC Dashboard is a great tool for exercising and learning about the capabilities of the DMCC service, quickly and easily. The DMCC Dashboard helps you design unified communications and contact center applications, prototype functionality and demonstrate your ideas to other interested parties.

So as a summary, here is a list of the top ten key facts about the DMCC Dashboard, and the underlying services and platforms it runs against:

1. Avaya Application Enablement (AE) Services provides connectivity between client applications and Avaya Communication Manager, Avaya's flagship IP telephony software platform.
2. The AE Services Device, Media, and Call Control (DMCC) service allows client applications to leverage Communication Manager's physical device, media, and first- and third-party call control capabilities.
3. The DMCC service provides three access methods for different programming languages: a Java API, a .NET API and an XML protocol description. The DMCC XML protocol description can be used for client applications written in any language, such as C or C++, which supports the sending and receiving of XML data over a network connection.
4. The DMCC Dashboard is designed specifically to allow users to exercise and observe just about all of the capabilities of the DMCC service. The DMCC Dashboard is an invaluable tool for developers and others who want to learn about and use the functionality that the DMCC service provides.

5. Although the DMCC Dashboard itself is written in .NET, the concepts and capabilities that it demonstrates apply equally to all of the DMCC service access methods and SDKs.
6. The DMCC Dashboard offers a context-aware interface, which enables additional device, media, and call control functionality only when pre-requisite steps (such as device registration) have been completed. This helps you to understand the order and structure of API calls that need to be made in your applications.
7. The DMCC Dashboard displays the underlying XML messages exchanged between itself and the DMCC service. This feature is particularly valuable for developers who wish to use the DMCC XML SDK because the messages provide a template for the XML that their client applications need to generate and handle.
8. The DMCC Dashboard can be used to demonstrate the following capabilities of the DMCC service: physical device (phone set) control, including first-party call control; server- and client-side media control for call recording and other media processing applications; and basic third-party call control.
9. The DMCC Dashboard can be used to prototype simple applications and pieces of functionality by stepping through the required API method calls and observing the results.
10. API method calls made from the DMCC Dashboard can be saved to a script and subsequently replayed to aid automated testing activities, such as network and regression testing.

Appendix

DevConnect Developer & Partner Program

As a communications application developer, you've already made an investment in the future of intelligent communications solutions for your business. We'd like to help you get more out of that investment, by providing you with the tools and resources to take your business to the next level.

Whether you are simply exploring how Avaya technology can create opportunities for your company and customers, or seeking additional technical know-how and information to aid in your current development efforts, a free DevConnect membership offers access to:

- Downloadable software development kits (SDKs) and client-side libraries aid in application development and integration.
- Step-by-step tutorials, in-depth Flash-based training courses, and on-demand webinars build skills within your technical community.
- Technical Frequently Asked Questions (FAQs) and community-supported forums assist in knowledge sharing.
- Sample Applications jump start your application development efforts by demonstrating how to use key APIs.
- Developer conferences, podcasts, newsletters and other developer communications provide ongoing technical awareness.
- Simulators* and remote labs aid in prototyping and proof-of-concept implementations with minimal up-front investments.
- Evaluation kits and developer editions for emerging Avaya products, allow you to explore new solutions that can give you a competitive advantage or help you deliver superior customer service.
- Interoperability Notes and DevConnect Member Application Notes created by our Solution and Interoperability Test Lab assist in deploying solutions.
- Opportunities to participate in technical beta programs give you a leg up on next generation application development activities.
- And much more…

* Additional media & support charges may apply

A sample listing of available DevConnect resources:

SDK & API DOWNLOADS AND DOCUMENTATION

Information on over 30 product interfaces, APIs and SDKs, including:

- AE Services, including: Telephony & System Management Web Services; DMCC SDKs for Java, .NET and XML; and JTAPI/TSAPI
- IP Telephone Push API and PushSDK
- SIP Enablement Services Personal Profile Manager web service
- Proactive Contact Agent & Event Services APIs
- Meeting Exchange Bridge Control API
- Interaction Center 7.1 Client SDK

SIMULATORS & REMOTE LABS

- Avaya IP Communication Development Environment (IPCoDE), featuring AE Services, Communication Manager & SIP Enablement Services
- Self Service Remote Lab, featuring Avaya Voice Portal
- AE Services & Communication Manager Remote Lab

TRAINING COURSES, TUTORIALS & TECHNICAL WEBINARS

- Over 25 hours of in-depth training courses on Avaya product interfaces, including Avaya Distributed Office, SIP, Interaction Center, Dialog Designer and Avaya IP Telephone APIs
- Step-by-step tutorials, including:
    - Setup, Application Initialization, and Event Monitoring using the AE Services DMCC Java SDK
    - Designing a Microsoft SQL database connector in Dialog Designer
    - Avaya IP Telephones Push and Browser Applications Setup
    - Over 15 on-demand Technical Webinars, including An Introduction to the Voice Portal and Dialog Designer, and Avaya Distributed Office and its CTI capabilities

TOOLS, EVALUATION KITS & DEVELOPER EDITIONS

- Dialog Designer
- DMCC Dashboard

SAMPLE APPLICATIONS

More than 25 sample applications (in addition to those provided with individual SDKs) for AE Services, Dialog Designer, Meeting Exchange, Proactive Contact, and SIP Enablement Services (SES).

REGISTER TODAY

Register today for your free DevConnect membership at www.avaya.com/devconnect

Application Enablement Services SDKs

Avaya Application Enablement (AE) Services is a software platform that provides connectivity between client applications and Avaya Communication Manager. AE Services includes an enhanced set of Application Programming Interfaces (APIs), client-side libraries, protocols, and web services that expose the capabilities of Communication Manager to application developers.

AE Services provides application developers with a variety of Software Development Kits (SDKs), offering differing levels of programmatic control, protocol support, and programming language support. The table below can help you determine which AE Services SDKs are most appropriate for your application development needs.

	Programming language/protocol				
Functionality	Java	C/C++	.NET	Any/XML	Web Services
Advanced third party call control	JTAPI SDK	TSAPI SDK			
Basic third party call control	DMCC Java SDK	DMCC XML SDK	DMCC .NET SDK	DMCC XML SDK	
Simple call creation and maintenance					Telephony SDK
Physical device control	DMCC Java SDK	DMCC XML SDK	DMCC .NET SDK	DMCC XML SDK	
Media control	DMCC Java SDK	DMCC XML SDK	DMCC .NET SDK	DMCC XML SDK	
Communication Manager system management					SMS SDK

AE Services SDK Selection Matrix

TSAPI, JTAPI and CVLAN SDKs

The TSAPI, JTAPI and CVLAN SDKs provide tools to help developers create applications that make use of the AE Services TSAPI (Telephony Services API) for Avaya Communication Manager, JTAPI (Java Telephony API) for Communication Manager and CVLAN (CallVisor LAN) interfaces, respectively. Each of these public interfaces enables access to the full complement of third party call control capabilities provided by Communication Manager. The interfaces are known collectively as the AE Services Computer Telephony Integration (CTI) APIs.

The following SDKs are available for the corresponding CTI APIs:

TSAPI SDK: for developing C and C++ applications. Windows and Linux versions of the TSAPI SDK are available.

JTAPI SDK: for developing Java applications. Windows and operating system independent versions of the JTAPI SDK are available.

CVLAN SDK: for developing applications that exchange low-level Adjunct Switch Application Interface (ASAI) messages with the AE Services server. Windows and Linux versions of the CVLAN SDK are available. ***CVLAN is an Avaya-specific protocol and is not intended for new application development.***

DMCC SDKs

The Device, Media, and Call Control (DMCC) SDKs provide tools to help developers make use of the AE Services DMCC APIs and protocols. The DMCC APIs and protocols enable applications to access to the first party physical device control, first party media control and basic third party call control capabilities of Communication Manager. The following DMCC SDKs are available for developers:

DMCC Java SDK: for developing Java applications.

DMCC XML SDK: for developing applications in any language that supports the sending and receiving of XML data over a network connection.

DMCC .NET SDK: for developing .NET applications.

Telephony Web Service SDK

The AE Services Telephony Web Service provides a high level interface to a subset of the third-party call control capabilities available on Avaya Communication Manager. The Telephony Web SDK comprises tools, including sample code, to help develop SOAP clients that include simple call creation and control.

System Management Service SDK

The System Management Service (SMS) is a web service provided by AE Services that exposes selected management features of Avaya Communication Manager. The web service enables SOAP clients to display, list, add, change and remove specific managed objects on Communication Manager.

The SMS web service SDK comprises tools, including sample code, to help develop web applications that include Communication Manager management capabilities.

For more information on the AE Services SDKs, see the individual fact sheets available for each SDK.

Getting started with the AE Services SDKs

All of the AE Services SDKs, with the exception of the TSAPI SDK, are available as free downloads from the Avaya DevConnect web portal (registration required). Gold and Platinum DevConnect members can order the TSAPI SDK via their procurement benefits; registered DevConnect members and other users should order the TSAPI SDK from an authorized Avaya Business Partner or Avaya Account Executive.

Avaya IP Communications Development Environment

The Avaya IP Communications Development Environment (Avaya IPCoDE) enables developers to test and debug IP communications applications under development, cost-efficiently and easily. Avaya IPCoDE comprises software-only, developer-oriented editions of Avaya Application Enablement (AE) Services, Avaya Communication Manager and Avaya SIP Enablement Services. The complete environment can be installed and run on a single machine.

Benefits of Avaya IPCoDE

The Avaya IP Communications Development Environment allows DevConnect members to:

- Reduce their up-front investment in Avaya products for debugging and unit testing IP communications applications under development.
- Speed up application development by providing an easily accessible debugging and unit testing environment on the developer's desktop.
- Validate IP communications applications in preparation for DevConnect compliance testing in the Avaya Solution and Interoperability Test Lab.
- Run sample applications, included in the appropriate SDKs and available for download from the DevConnect web portal, to gain an appreciation of the capabilities of Avaya solutions.
- Use tools such as the AE Services DMCC Dashboard, TSAPI Exerciser and JTAPI Exerciser to expedite and learn about the capabilities of the corresponding services, and aid application development. These tools are included in the appropriate SDKs.
- Set up an environment for other Avaya products with a dependency on Avaya Communication Manager, such as Avaya Voice Portal.

Debugging and Testing Applications

Avaya IPCoDE is ideally suited for debugging and testing applications under development, including:

- AE Services applications developed against the AE Services APIs, protocol descriptions and web services that enable access to the capabilities of Communication Manager. These include applications that use the Device, Media, and Call Control (DMCC) service, TSAPI service, JTAPI service, Telephony Web Service and System Management Service (SMS) Web Service.
- Avaya Communication Manager operations, administration, maintenance and provisioning applications.
- Avaya SIP-enabled applications, including:
    - Applications that expose the capabilities of Communication Manager to Avaya SIP telephones.
    - Applications that incorporate SIP-based Presence and Instant Messaging with Avaya IP Softphone, Avaya IP Agent, Avaya one-X™ Desktop, and Avaya one-X™ Deskphones.
- Avaya SIP Personal Profile Manager (PPM)-based applications, utilizing web services exposed by Avaya SIP Enablement Services for user-specific information.
- Avaya IP telephone applications that leverage the capabilities exposed by the Avaya IP telephone APIs, such as the Push API.

In addition, the environment can be used to debug and test applications that support various types of phones, including SIP and H.323 IP telephones.

- Note: While Avaya IPCoDE provides fully-featured editions of the software, certain functional limitations exist, including some media processing capabilities. See the DevConnect web portal for additional details.

Obtaining the Avaya IP Communications Development Environment

Information on how to request your copy of Avaya IPCoDE, including details of the release-cycle, licensing and pricing, is available on the DevConnect web portal (registration required). Procurement discounts are available to DevConnect Gold and Platinum members.

Technical Summary

IP Phone Connectivity

The Avaya IP Communications Development Environment provides connectivity for up to 10 IP desktop telephones or soft phones.

Network Options

To provide flexibility of use, Avaya IPCoDE can be configured so that it can only be accessed from the host machine, or bridged to a private or public network.

Installation Requirements

To install and use Avaya IPCoDE, developers need:

- A desktop PC or server running Microsoft Windows 2000, 2003 or XP operating system. See the DevConnect web portal for details of the minimum requirements for the machine processor, RAM and hard disk space.
- A VMware environment installed on the PC or server.
- A DVD drive from which to install the Avaya IPCoDE environment.
- Adobe Acrobat Reader.

A dual monitor system is recommended as an effective method for monitoring the many simultaneous points of interaction and information exchange between the Avaya IPCoDE elements and the applications under development.

Additional Resources for Developers

DevConnect members who require access to media gateway and other IP Communication features not supported by Avaya IPCoDE can also use the free AE Services & Communication Manager Remote Lab in support of development, testing and pre-compliance activities

Mastering Device, Media and Call Control

An Avaya DevConnect Publication.

Email devconnect@avaya.com with comments or questions regarding this book.

Speed IP Communications Applications Development

If you are in the business of specifying, designing or developing advanced IP communications applications, use the Avaya DMCC Dashboard to simplify and accelerate the development process:

- Understand device, media and call control functionality available to your applications
- Rapidly prototype unified communications and contact center applications, without writing any code
- Test and troubleshoot application logic and structure with script recording and replay

This book is your complete guide to the Avaya DMCC Dashboard, a purpose-built tool for learning the Device, Media and Call Control (DMCC) Service API for Avaya Application Enablement Services. Step-by-step overviews of the features, usage and benefits of the DMCC Dashboard make it easy to master these capabilities for your own advanced IP communications applications.

Learn More

Visit www.avaya.com/devconnect to learn more about the Avaya DevConnect Program and tools such as the DMCC Dashboard.

About Avaya

Avaya delivers Intelligent Communications solutions that help companies transform their businesses to achieve market-place advantage. More than 1 million businesses worldwide, including more than 90 percent of the FORTUNE 500®, use Avaya solutions for IP Telephony, Unified Communications, Contact Centers and Communications Enabled Business Processes. Avaya Global Services provides comprehensive service and support for companies, small to large. For more information visit the Avaya Web site: **http://www.avaya.com.**

AVAYA

INTELLIGENT COMMUNICATIONS

avaya.com

© 2009 Avaya Inc. All Rights Reserved.
Avaya and the Avaya Logo are trademarks of Avaya Inc. and may be registered in certain jurisdictions.
All trademarks identified by ®, TM or SM are registered marks, trademarks, and service marks, respectively, of Avaya Inc., with the exception of FORTUNE 500 which is a registered trademark of Time Inc. All other trademarks are the property of their respective owners.
1/09 MIS4040-DEV