

Programming Converged Networks

PROGRAMMING CONVERGED NETWORKS

CALL CONTROL IN JAVA, XML, AND
PARLAY/OSA

Ravi Jain

DoCoMo Communications Laboratories USA

John-Luc Bakker

Farooq Anjum

Telcordia Technologies

 **WILEY-
INTERSCIENCE**

A JOHN WILEY & SONS, INC. PUBLICATION

This book is printed on acid-free paper. ∞

Copyright © 2005 by John Wiley & Sons. All rights reserved.

Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (508) 750-8400, fax (508) 750-4744. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, fax (201) 748-6008, E-mail: PERMREQ@WILEY.COM.

For ordering and customer service, call 1-800-CALL-WILEY.

Library of Congress Cataloging-in-Publication Data is available.

ISBN 0-471-26801-1

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

To Meera and Kaahini

— R.J.

To Karin, Thijmen, and Quintin

— J.-L.B.

To My Parents

— F.A.

CONTENTS

List of Figures	xv
Preface	xix
Acknowledgments	xxiii
1. Introduction	1
1.1 Sessions and Call Control	3
1.2 Programmability and APIs	5
1.3 How This Book Is Organized	6
1.4 Relevant Industry Fora	7
1.5 Concluding Remarks	8
1.6 To Explore Further	9
2. The Telephone Network, Converged Networks, and Programmability	11
2.1 Evolution of the PSTN	11
2.1.1 Circuit and Packet Switching	11
2.1.2 Switches and Signaling	12
2.1.3 Switch-Based Services	14
2.1.4 The (Advanced) Intelligent Network	15
2.1.5 Converged Networks	16
2.1.6 A Glance at Signaling Protocols: ISUP and SIP	19
2.2 Programmability in AIN and Converged Networks	23
2.2.1 Limitations of AIN	23
2.2.2 Programmability in Converged Networks	23
2.3 APIs Versus Protocols	24
2.3.1 The Difference Between APIs and Protocols	24
2.3.2 APIs Versus Protocols for Programmability	26
2.4 Motivating Services for API Definitions	27
2.4.1 Voice Virtual Private Network	28
2.4.2 Voice-Activated Dialing	29
2.4.3 Click-To-Dial	30

2.5	Running Examples Used in This Book	32
2.5.1	Toll-Free Calling	32
2.5.2	Internet Call Waiting	33
2.6	Concluding Remarks	33
2.7	To Explore Further	34
3.	Basic Concepts and Design Issues	35
3.1	Introduction	35
3.2	Basic Terminology	35
3.2.1	Applications and Services	35
3.2.2	Service and Network Providers	36
3.2.3	Users and Subscribers	37
3.2.4	Switches and Switching Centers	37
3.3	A Simple Model of Call Processing	39
3.4	Basic Model of Programmability	43
3.5	First- and Third-Party Call Control	44
3.6	Call Models and APIs	45
3.7	Specification Language	47
3.7.1	Specification Languages	47
3.7.2	Choosing a Language	48
3.8	Full and Half-Call Models	51
3.9	Symmetric and Asymmetric Call Models	52
3.10	Network and Application Views of Call Processing	53
3.11	Network-Edge and Network-Core APIs	54
3.12	API Extensibility	55
3.13	Feature Interaction	56
3.14	Concluding Remarks	58
3.15	To Explore Further	58
4.	The Advanced Intelligent Network	59
4.1	Introduction	59
4.1.1	History	59
4.1.2	Architecture	60
4.2	SS7	61
4.2.1	SS7 Elements	61
4.2.2	SS7 Protocols	63
4.3	Core AIN and IN Aspects	64
4.3.1	AIN/IN Principles	64
4.3.2	IN Processing Model	66
4.3.3	Basic Call State Models	67

4.4	ITU-T IN Aspects	71
4.4.1	ITU-T IN Architecture and Capability Sets	71
4.4.2	Service-Independent Building Blocks, Services, and Service Features	73
4.5	Wireless Intelligent Networks (WIN)	74
4.5.1	History	74
4.5.2	WIN Architecture	75
4.6	Customized Applications for Mobile Enhanced Logic (CAMEL)	77
4.7	AIN Examples	78
4.7.1	Toll-Free Calling	78
4.7.2	Internet Call Waiting	79
4.7.3	900 Call Restriction with PIN Override	80
4.8	Concluding Remarks	81
4.9	To Explore Further	82
5.	The Java Telephony API	83
5.1	Introduction	83
5.1.1	What Is Computer Telephony Integration (CTI)?	83
5.1.2	Why JTAPI?	84
5.1.3	History of JTAPI	85
5.2	JTAPI Basics	85
5.3	Basic Call Scenarios	88
5.3.1	Two-Party Call	88
5.3.2	Two-Party Call with Multiple Terminals	89
5.3.3	Three-Party Call	90
5.4	The JTAPI Peer and JTAPI Packages	90
5.4.1	Core	92
5.4.2	Call Control	94
5.4.3	Call Center	97
5.4.4	Mobile	98
5.4.5	Phone	98
5.4.6	Media	98
5.4.7	Private Data	98
5.5	JTAPI Examples	98
5.5.1	Provisioned Call Forwarding Application	99
5.5.2	Dynamic Call Forwarding Application	101
5.6	Distributed JTAPI	103
5.6.1	Full Call Model	104
5.6.2	Half Call Model	105
5.7	Concluding Remarks	106
5.8	To Explore Further	106

6. JAIN Call Control: JCC and JCAT	107
6.1 Introduction	107
6.2 Background	107
6.3 JCC and Signaling Protocols	110
6.4 JCC and Application-Level Facilities	110
6.5 Call Control in JAIN	111
6.6 JCC and JCAT Service Drivers	113
6.6.1 JCC Service Drivers	113
6.6.2 JCAT Service Drivers	114
6.7 Components of the JCC/JCAT API	115
6.7.1 Basic Components	115
6.7.2 Advanced Call Control Objects	116
6.7.3 Basic API Patterns: Listeners and Factories	116
6.7.4 Event and Listener Inheritance Diagrams	117
6.8 More About Java Call Control	118
6.8.1 JccProvider	118
6.8.2 JccCall	119
6.8.3 JccConnection	121
6.8.4 JccAddress	124
6.8.5 Event Filters	125
6.9 Java Call Control EXTensions	125
6.9.1 JcatProvider	125
6.9.2 JcatCall	126
6.9.3 JcatConnection	126
6.9.4 JcatAddress	127
6.9.5 JcatTerminal	127
6.9.6 JcatTerminalConnection	127
6.10 JCC Call Flows	128
6.10.1 First-Party Call	131
6.10.2 A Call Logging Application	133
6.11 Running Examples Using JCC and JCAT	138
6.11.1 Toll-Free Call Application	138
6.11.2 Internet Call Waiting Application	142
6.12 API and Protocols	145
6.12.1 JCAT Merge Call Mapping to SIP	146
6.13 Relationship of JCC/JCAT to JTAPI and Parlay APIs	147
6.14 Concluding Remarks	148
6.15 To Explore Further	148
7. The Parlay/OSA API	151
7.1 Introduction	151

7.2	History and Background	151
7.2.1	The JWG Process and Technology Mappings	154
7.2.2	The Influence of TINA-C	155
7.2.3	The TINA Business Model	156
7.3	Parlay Architecture	157
7.3.1	Registering and Using Parlay Services	158
7.4	Overview of Parlay APIs	160
7.4.1	Framework	160
7.4.2	Call Control APIs	163
7.4.3	User Interaction	163
7.4.4	Mobility	164
7.4.5	Terminal Capabilities	164
7.4.6	Data Session Control	164
7.4.7	Generic Messaging Service	165
7.4.8	Connectivity Manager	165
7.4.9	Account Management	165
7.4.10	Charging API	166
7.4.11	Policy Management Service	166
7.4.12	Presence and Availability Management API	167
7.5	Design Patterns and Naming Conventions	167
7.5.1	Design Patterns and Conventions	167
7.5.2	Naming Conventions	169
7.6	Call Control APIs	171
7.6.1	Generic Call Control API	171
7.6.2	Multiparty Call Control	174
7.6.3	Multimedia Call Control	178
7.6.4	Conference Call Control	179
7.7	Steps in the Early Life of a Parlay Application	181
7.7.1	How the Application Accesses the Initial Interface	182
7.7.2	Authenticating the Framework and Application	183
7.7.3	Requesting Access to the Service Discovery Interface	185
7.7.4	Discovering Parlay Services	186
7.7.5	Signing the Service Agreement	187
7.8	Steps in the Early Life of a Parlay Service	189
7.8.1	How the Service Accesses the Initial Interface	191
7.8.2	Authenticating the Framework and Service	191
7.8.3	Requesting Access to the Service Registration Interface	192
7.8.4	Registering a Parlay Service	192
7.9	The Usage Session: Multimedia Call Control	193

7.10	Running Examples in Parlay	196
7.10.1	Toll-Free Application	196
7.10.2	Internet Call Waiting Application	202
7.11	Concluding Remarks	205
7.12	To Explore Further	205
8.	Detailed API Design Issues	207
8.1	Introduction	207
8.2	Synchronous Versus Asynchronous Calls	207
8.3	FSM Inheritance Considerations	208
8.4	Object Mutability	210
8.4.1	Multiple Sessions per Object	211
8.5	Callbacks and Event Listeners	212
8.6	Handling Events	212
8.7	Event and Listener Hierarchy	213
8.8	Using Interfaces Versus Using Classes	214
8.9	Bootstrapping, Factories, and Peers	216
8.10	To Explore Further	217
9.	XML Programmability: PINT, SPIRITS, JAIN SCML, and Parlay X	219
9.1	Introduction	219
9.2	PINT and SPIRITS	221
9.2.1	PSTN/Internet Interworking Protocol (PINT)	222
9.2.2	Service in the PSTN/IN Requesting InTernet Services (SPIRITS)	223
9.2.3	Authorization, Authentication, and Brokering	225
9.3	Service Creation Markup Language	227
9.3.1	Developing and Deploying an SCML Script	228
9.3.2	An Example Using SCML	230
9.3.3	Using XSL to Generate Scripts	231
9.4	Parlay X Web Services	231
9.5	Concluding Remarks	234
9.6	To Explore Further	234
10.	Concluding Remarks and a Look at the Future	235
10.1	Problems with Existing APIs	236
10.1.1	Network Intelligence Model	236
10.1.2	API Complexity and Overhead	236
10.1.3	Specification Rigor	237
10.1.4	Security	238
10.1.5	Support for Mobile Wireless Networks	239

10.1.6 Support for Alternative Charging and Billing Models	240
10.1.7 The Bottom Line: The Elusive “Killer App”	241
10.2 A Speculative Look at the Future	242
10.2.1 Scenario 1: APIs Everywhere	242
10.2.2 Scenario 2: No APIs and No Call Models	242
10.2.3 Scenario 3: APIs and Call Models, but No Standard	243
10.2.4 Scenario 4: Other Approaches to Programmability	243
10.3 Concluding Remarks	244
Acronyms	247
References	253
Index	261

LIST OF FIGURES

Figure 1–1.	A simplified view of the parties using an API.	2
Figure 1–2.	API specification and implementation.	6
Figure 2–1.	PSTN circuit-associated signaling system.	12
Figure 2–2.	PSTN common-channel signaling (CCS) system.	13
Figure 2–3.	AIN architecture.	16
Figure 2–4.	A high-level view of the converged network architecture.	17
Figure 2–5.	Converged network architecture.	18
Figure 2–6.	ISUP call initiation.	20
Figure 2–7.	ISUP call answer.	20
Figure 2–8.	ISUP call release.	21
Figure 2–9.	Basic call setup and teardown using SIP.	22
Figure 2–10.	APIs and protocols.	25
Figure 2–11.	Virtual Private Network architectural scenario.	28
Figure 2–12.	VAD call flow.	30
Figure 2–13.	CTD call flow scenario.	31
Figure 3–1.	A simplified view of the PSTN switching hierarchy.	38
Figure 3–2.	Objects in JTAPI core call model.	40
Figure 3–3.	JTAPI connection object FSM.	41
Figure 3–4.	State transitions in the setup and teardown of a JTAPI call.	42
Figure 3–5.	Basic model for programming a new service.	43
Figure 3–6.	Example of an application using JTAPI to originate a call.	46
Figure 3–7.	Example of pre- and post-conditions.	50
Figure 3–8.	Application-view and network-view API scenarios.	54
Figure 3–9.	JTAPI Call Control extension Connection object FSM.	56
Figure 3–10.	Pre- and post-conditions for <code>Connection.answer()</code> .	57
Figure 4–1.	Generic AIN architecture.	61
Figure 4–2.	Bell System SS7 Network.	62
Figure 4–3.	SS7 protocol stack.	63
Figure 4–4.	First intelligent network (IN/1) architecture versus AIN architecture.	65
Figure 4–5.	Service processing before IN (from: ITU-T Recommendation Q.1206, Fig. 15).	66
Figure 4–6.	IN service processing model (from: ITU-T Recommendation Q.1206, Fig. 16).	67
Figure 4–7.	Simplified Originating Basic Call State Model based on AIN Issue 8.	68
Figure 4–8.	Terminating Basic Call State Model based on AIN Issue 8.	70
Figure 4–9.	ITU-T IN reference architecture and functional elements.	72

xvi LIST OF FIGURES

Figure 4–10.	Possible WIN architecture and functional elements.	75
Figure 4–11.	Toll-free calling AIN-supported service.	79
Figure 4–12.	Internet Call Waiting AIN supported service.	80
Figure 4–13.	900 Call Restriction with PIN override AIN supported service.	81
Figure 5–1.	Basic components of a CTI system.	84
Figure 5–2.	JTAPI Call Model.	86
Figure 5–3.	Two-party call with local addresses.	89
Figure 5–4.	A two-party call with local addresses and with multiple terminals for the destination party.	90
Figure 5–5.	A three-party call on a single platform with multiple terminals being used.	91
Figure 5–6.	JTAPI implementation.	91
Figure 5–7.	Sequence diagram for a JTAPI call setup.	99
Figure 5–8.	JTAPI provisioned call forwarding using Call Control package.	100
Figure 5–9.	JTAPI Provisioned Call Forwarding initialization and usage.	102
Figure 5–10.	JTAPI ICW using Call Center package.	103
Figure 5–11.	JTAPI ICW initialization.	103
Figure 5–12.	JTAPI ICW usage.	104
Figure 5–13.	A two-party call with remote address and the full call model.	105
Figure 5–14.	A two-party call with remote address and the half call model.	105
Figure 6–1.	Basic JAIN architecture.	108
Figure 6–2.	JAIN logical architecture (a.k.a. “religious diagram”).	109
Figure 6–3.	Object model of a two-party call.	116
Figure 6–4.	JCAT object model for a two-party call.	117
Figure 6–5.	API programming pattern using Java Listeners.	117
Figure 6–6.	JCC Event and Listener inheritance diagrams.	118
Figure 6–7.	JccProvider FSM.	119
Figure 6–8.	JCC Call object FSM.	120
Figure 6–9.	JCC Connection object FSM.	122
Figure 6–10.	JCAT TerminalConnection object FSM.	128
Figure 6–11.	First-party call placed with JCC.	129
Figure 6–12.	Application monitoring and logging call activity using JCC.	130
Figure 6–13.	Call logging application.	137
Figure 6–14.	JCC Toll-Free call.	139
Figure 6–15.	JCC Toll-Free call application initialization.	140
Figure 6–16.	JCC Toll-Free call usage.	141
Figure 6–17.	JCC Toll-Free call mapped to SIP.	141
Figure 6–18.	JCC Internet Call Waiting.	142
Figure 6–19.	JCC Internet Call Waiting initialization.	143
Figure 6–20.	JCC Internet Call Waiting usage.	144
Figure 6–21.	JCC Internet Call Waiting mapped to SIP.	145
Figure 6–22.	JCC/JCAT object collaboration diagram.	146
Figure 6–23.	JCAT Merge Call mapped to SIP.	147
Figure 7–1.	The Parlay APIs in the converged network architecture.	152
Figure 7–2.	Relation of API standardization bodies.	153

Figure 7-3.	The Parlay and 3GPP OSA relationship.	154
Figure 7-4.	Parlay technology realizations.	155
Figure 7-5.	TINA business roles and reference points.	156
Figure 7-6.	General Programmable Gateway architecture.	158
Figure 7-7.	High-level overview of registering and using Parlay services.	159
Figure 7-8.	Parlay/OSA Call Control inheritance diagram.	163
Figure 7-9.	FSM of IpCallControlManager.	172
Figure 7-10.	FSM of IpCall.	173
Figure 7-11.	FSM of the IpMultiPartyCallControlManager.	175
Figure 7-12.	FSM of the IpMultiPartyCall.	176
Figure 7-13.	Originating FSM of IpCallLeg as found in [3GPP 2002f].	177
Figure 7-14.	Terminating FSM of IpCallLeg as found in [3GPP 2002g].	178
Figure 7-15.	The application accesses the initial interface.	183
Figure 7-16.	The application authenticates the Framework.	184
Figure 7-17.	Mutual authentication.	185
Figure 7-18.	Request access and service discovery.	186
Figure 7-19.	Gaining access to the service discovery interface.	187
Figure 7-20.	Multimedia Call Control service discovery.	188
Figure 7-21.	Mutual signing of the service agreement.	189
Figure 7-22.	Application signs service agreement.	190
Figure 7-23.	Service gets access to initial interface.	191
Figure 7-24.	Service and Framework mutually authenticate.	192
Figure 7-25.	Service retrieves registration interface.	192
Figure 7-26.	Service registration.	193
Figure 7-27.	Service registration code.	194
Figure 7-28.	MMCCS' interfaces and cardinality.	194
Figure 7-29.	Registering for a static notification.	195
Figure 7-30.	MMCC application.	196
Figure 7-31.	Receiving a disposition request.	197
Figure 7-32.	Parlay MPCCS Toll-Free application.	197
Figure 7-33.	Parlay MPCCS Toll-Free application partial initialization.	199
Figure 7-34.	Parlay MPCCS Notification registration utility method.	200
Figure 7-35.	Parlay MPCCS Toll-Free application usage.	201
Figure 7-36.	Parlay MPCCS ICW.	202
Figure 7-37.	Parlay MPCCS ICW partial initialization.	203
Figure 7-38.	Parlay MPCCS ICW usage.	204
Figure 8-1.	FSM of the base class.	209
Figure 8-2.	Proposed FSM of the child class.	210
Figure 8-3.	JccConnection interface.	215
Figure 8-4.	JccConnection abstract class.	215
Figure 8-5.	Java class extending the parent class and implementing an interface.	215
Figure 8-6.	One option when using the Factory pattern for bootstrapping.	216
Figure 8-7.	The JccPeer factory pattern.	217
Figure 9-1.	PINT and SPIRITS.	221
Figure 9-2.	Click-To-Dial example.	223
Figure 9-3.	SPIRITS architecture in conjunction with programmability architecture.	224

xviii LIST OF FIGURES

Figure 9–4.	Example of a SPIRITS INVITE message.	226
Figure 9–5.	Script interpreter integrated with the programmable gateway or with the application server.	228
Figure 9–6.	Script creation, deployment, and execution.	229
Figure 9–7.	Example SCML script: Call Forwarding on Busy.	230
Figure 9–8.	Example SCML scripts: SMS on Busy Subscriber.	231
Figure 9–9.	Using XSLT to generate SCML.	232
Figure 9–10.	Relationship between Parlay X and Parlay APIs.	233
Figure 10–1.	Separating Users, Terminals, and Addresses.	239
Figure 10–2.	Performance versus flexibility tradeoff.	244

■ PREFACE

WHY THIS BOOK?

When we first started working in the area of programmability for converged networks, in the late 1990s, we found we had to wade through manuals, specifications, design documents, a few research papers, and even code, to understand the subject. It was tough going.

In fact, even the definition of converged network—a communications network that seamlessly integrates packet, circuit, wired, and wireless networks—was still emerging, and often it went by other names, such as Next Generation Network. (We found this last name a little confusing, since, after all, every new network architecture is a “next generation” network.)

As we continued developing designs and prototypes of our ideas for APIs, writing our own research papers, and then participating in design teams as well as attending industry forums, things often just seemed to get more confusing, not less. (They did not get much clearer when we started leading the design teams and chairing the industry working groups ourselves.)

But we did come to one firm conclusion: There ought to be a book.

Well, this is supposed to be that book. Hopefully it will provide, at least, a single place to look for information on network programming APIs for people new to the area as well as those who already have experience in it.

If not, perhaps it will be a convenient coaster or door stop.

WHAT'S IN THIS BOOK?

The book does not attempt to cover all aspects of converged network APIs. That would double its size without, in our opinion, adding much needed insight. Instead, it focuses on what we consider to be the core technical issue in such APIs, and usually the most complex: call control.

Call control is the process of setting up, manipulating, and tearing down communications sessions. The name derives from the telephony world, but do not let that turn you off. In this book a call means, in general, a multimedia communication session involving multiple parties communicating over the Internet, the telephone network, or a converged network. We think that is pretty interesting given how prevalent such sessions are becoming in our daily lives. Also, call control forms

the basis for innovative new communications services and applications that add value beyond transporting bits from one part of the world to another.

Section 1.3 gives a brief outline of the book, and we will not repeat that here. Suffice it to say that we try to present, in Chapters 1–3, the basic concepts and background, including the fairly simple and logical ideas underlying call processing and services.

The next four chapters (Chapters 4–7) describe the logical progression of network programmability, starting with Intelligent Network concepts, continuing through the innovations in the enterprise telephony world represented in the JTAPI API, and on, to the JAIN and Parlay/OSA APIs developed by the telecommunications industry. With this under our belt, in Chapter 8 we scratch the surface of advanced design issues when developing an API. We then present a summary of the exciting developments of network programming using XML.

Throughout the book we use running examples and code fragments as well as call flows to help make the material concrete.

Finally, by the last chapter (Chapter 10) we feel we have earned the right to get up on a soapbox and to make some off-the-wall predictions.

WHO SHOULD READ THIS BOOK?

Well, everybody, of course. Especially if you pay full price at a bookstore. (Just kidding.)

The book is intended for programmers, engineers, researchers, and managers who deal with converged networks and want to know how they can be programmed using industry standard APIs. It is also suitable as a reference for graduate students and technical professionals new to the industry or this area.

In general, we assume that you have some familiarity with object-oriented programming and design, at least enough to be dangerous. Some knowledge of Java is also helpful, along with a smattering of XML, as some of the examples use these languages.

A healthy ability to tolerate acronyms is, unfortunately, also required. Acronyms are a pestilence in the telecommunications and Internet industries, and there seems to be no way of escaping them. (By the way, converged networks do not seem to result in fewer acronyms, just converged acronyms.) We have tried to help by defining acronyms when they are first used and providing a list at the back of the book. But we can only do so much. The rest is up to you: The next time you see someone defining a new acronym for their project just because it sounds kind of cool, tell them politely, but firmly, to stop.

FINALLY . . .

This is an exciting time in the world of technology, and especially communications. (We are geeks enough to really believe that.) We hope the book helps to illuminate the interesting and intriguing area of programming converged communications

networks, which has the potential to result in useful, innovative, and entertaining new services for consumers.

The authors' website for this book, with more information and for sending us comments, is <http://www.research.telcordia.com/ProgrammingConvergedNetworks.html>. Be in touch!

RAVI JAIN
jain@docomolabs-usa.com

JOHN-LUC BAKKER
jbakker@telcordia.com

FAROOQ ANJUM
fanjum@telcordia.com

ACKNOWLEDGMENTS

We wish we could begin by saying “We would like to thank all the little people who...”, just as they do at the Oscars. But in this case we are the little people.

However, even little people need help. Lots of help.

Many thanks are due to:

the members of the JAIN JCC/JCAT, JAIN SCML, JWG, Parlay X and Parlay ETS working groups, who we worked with as some of the APIs described in this book were developed.

the Sun Microsystems JAIN team, including Margaret Nilson, Doug Tait, Gary Bruce, Steven Grover, Phelim O’Doherty, and Rob Goedman for being such a smart and fun crew to work with.

the Ericsson guys: Lukas Klostermann, Ard-Jan Moerdijk, and Erik van der Velden for their perseverance.

our excellent colleagues at Telcordia Applied Research in the ChaiTime project, when we got started in this area, especially Paolo Missier and Raman Shastry. Also Adalberto Zordan and Francesco Caruso.

Jim Garrahan and Surinder Jain of Telcordia Professional Services for many useful pointers.

Pat Donnelly of Telcordia Software Systems for championing open APIs in the face of much skepticism.

Several people reviewed chapters of this book, and provided many useful comments: Jim Garrahan and Surinder Jain (Telcordia), Rich Penanga (Avaya), Musa Unmehopa (Lucent) and Ard-jan Moerdijk (Ericsson), David Ferry and David Page (OpenCloud), Phelim O’Doherty (Sun), Raman Shastry (Leapstone), and John Wullert (Telcordia). Many thanks to all, and of course any mistakes remaining are our own.

Thanks are due to Minoru Etoh at DoCoMo USA Labs for supporting this project. We also thank Jan Clayton, Sarah Hitzeman and their colleagues at Expert Support for their expert support. Finally, we thank Val Moliere, Kirsten Rohstedt, Christine Punzo, and their colleagues at Wiley for their incredible patience.

