IBM

Mobile Commerce Solutions Guide

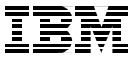
using WebSphere Commerce Suite V5.1

- IBM m-commerce architecture and functionality
- Design and development guidelines for m-commerce
- Sample code for WAP WML, HDML, Palm, and WTP

John Ganci
David Barker
Masaaki Ishibashi
Peter Kovari
Rodrigo Magalhaes
Giuseppe Plagenza
Masaaki Saitoh

Redbooks

ibm.com/redbooks



International Technical Support Organization

Mobile Commerce Solutions Guide using WebSphere Commerce Suite V5.1

July 2001

SG24-6171-00

Take Note! Before using this information and the product it supports, be sure to read the general information in “Special notices” on page 375.

First Edition (July 2001)

This edition applies to WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000, and WebSphere Commerce Studio V5.1, Professional Developer’s Edition for Windows NT and Windows 2000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Preface xi
The team that wrote this redbook xii
Special notice xiv
IBM trademarks xiv
Comments welcome xv

Part 1. Introduction to mobile commerce 1

Chapter 1. Introduction 3
1.1 Mobile commerce overview 5
1.1.1 m-commerce defined 5
1.1.2 Key objectives of m-commerce 5
1.1.3 Mobile devices 6
1.1.4 Challenges 7
1.2 m-commerce opportunities and market drivers 7
1.2.1 m-commerce opportunities 7
1.2.2 Mobile market penetration 8
1.2.3 Technology 12
1.2.4 New business models 12
1.2.5 Customer needs 14
1.3 IBM m-commerce 14
1.3.1 m-commerce using WebSphere Commerce Suite V5.1 15
1.4 Structure of this redbook 16

Chapter 2. Wireless technologies 19
2.1 Wireless networks 20
2.1.1 Mobile communications network history 20
2.1.2 GSM 24
2.1.3 GPRS 32
2.1.4 Mobitex 36
2.1.5 CDPD 38
2.1.6 PDC 39
2.1.7 IMT-2000 39
2.1.8 Wireless LANS 40
2.2 Wireless protocols 41
2.2.1 HTTP protocol 41
2.2.2 WAP 42
2.2.3 i-mode 46
2.2.4 Web Clipping 52

2.2.5 Short Message Service (SMS) 54
2.3 Mobile devices 57
2.3.1 Mobile phones 57
2.3.2 Wireless PDAs 58
2.3.3 Wireless laptops 62
2.3.4 Mobile device pros and cons 62
2.4 Content and markup languages 64
2.5 Wireless service providers 67
2.6 Next-generation technologies 67

Chapter 3. m-commerce development methodology 71
3.1 Understanding WCS V5.1 72
3.1.1 WCS V5.1 systems architecture and programming model 72
3.1.2 Session management 72
3.2 Market study of the customer environment 74
3.3 Existing or new site 75
3.4 Customer requirements 75
3.5 m-commerce implementation approaches 76
3.5.1 m-commerce direct 76
3.5.2 m-commerce using WTP - application runtime topology 77
3.5.3 Guidelines for selecting the implementation approach 78
3.6 Application design considerations 81
3.7 Application development 81
3.8 Testing the m-commerce application 83

Chapter 4. m-commerce features and functionality in WCS V5.1 85
4.1 WCS V5.1 m-commerce enablement overview 86
4.2 WCS V5.1 m-commerce enablement features 87
4.2.1 PvC adapter framework 87
4.2.2 PvC commands 90
4.2.3 PvC data beans 91
4.3 Session control 92
4.3.1 Unique identifier 92
4.3.2 PvC adapter 92
4.4 Device control 94
4.4.1 Differences between mobile devices 94
4.4.2 WCS V5.1 functionality for device differences 96
4.5 Security 97
4.5.1 Logon timeout 97
4.5.2 Restricted command execution 98
4.5.3 User registration mode 99
4.6 URL buffering 101
4.6.1 PVCBufferUrl command b_new mode 102

4.6.2 PVCBufferUrl command b_update mode 103
4.6.3 PVCBufferUrl command b_exec mode 104

Chapter 5. IBM wireless middleware 107
5.1 IBM WebSphere Everyplace Suite (WES) 108
5.1.1 WES overview 108
5.1.2 Considerations for m-commerce 112
5.1.3 Where to find more information 115
5.2 IBM WebSphere Transcoding Publisher (WTP) 115
5.2.1 WTP overview 116
5.2.2 Considerations for m-commerce 122
5.2.3 Where to find more information 123
5.3 IBM Everyplace Wireless Gateway (EWG) 123
5.3.1 EWG overview 124
5.3.2 Considerations for m-commerce 125
5.3.3 Where to find more information 125
5.4 IBM MQSeries Everyplace (MQe) 125
5.5 IBM DB2 Everyplace 126
5.6 IBM Mobile Connect 127

Chapter 6. m-commerce payment solutions 129
6.1 Payment technologies overview 130
6.1.1 Payment solutions for PC browser clients 130
6.1.2 Payment solutions for mobile devices 131
6.2 IBM WebSphere Payment Manager 136
6.2.1 WebSphere Payment Manager overview 136
6.2.2 Supported payment methods 137

Part 2. Setting up your m-commerce environment 139

Chapter 7. m-commerce runtime environment 141
7.1 WCS V5.1 runtime environment installation 142
7.1.1 ITSO test runtime environment 142
7.1.2 WCS V5.1 high-level installation steps 144
7.2 WTP V3.5 MIME filter installation 146
7.3 Configuring WTP V3.5 as a MIME filter 149
7.4 WCS V5.1 messaging configuration 155
7.4.1 Configure the WCS V5.1 messaging services 155
7.4.2 Define the message content 156
7.5 Where to find more information 157

Chapter 8. m-commerce development environment 159
8.1 Development environment 160
8.1.1 ITSO development environment 161

8.1.2 Development environment high-level installation steps 162
8.2 WebSphere Transcoding Toolkit. 163
8.2.1 Transform Tool 163
8.2.2 Request Viewer. 164
8.2.3 Profile Builder 165
8.2.4 Snoop MEGlet. 166
8.3 Test environments and tools 167
8.3.1 Toolkits and simulators 167
8.3.2 Real wireless hardware - intranet testing 170
8.3.3 Real wireless hardware - Internet testing 171
8.3.4 Everyplace Wireless Gateway installation 172
8.4 Where to find more information 177

Chapter 9. m-commerce sample store and sample code 179
9.1 Download sample store and sample code 180
9.2 Sample store overview 181
9.2.1 PvC Fashion sample store 182
9.2.2 Web Fashion sample store 183
9.3 Deploy sample store to WCS runtime 183
9.3.1 Create a store template. 184
9.3.2 Create a store from a template 186
9.3.3 Publish the PvC Fashion sample store from Store Services. 187
9.3.4 Create an alias for the store 188
9.3.5 Deploy PvC adapter and data beans 189
9.3.6 Configure content management 192
9.3.7 Verify PvC Fashion sample store 193
9.4 Prepare the development environment 195

Part 3. m-commerce direct implementation 197

Chapter 10. m-commerce direct design and development process. . . . 199
10.1 m-commerce direct application design guidelines. 200
10.2 m-commerce direct development process 203

Chapter 11. Creating and deploying a PvC adapter. 205
11.1 Creating a PvC adapter 206
11.1.1 Create a project in VisualAge for Java (VAJ) 207
11.1.2 Create a package in VAJ 207
11.1.3 Identify the device type 208
11.1.4 Create an adapter class 210
11.1.5 The checkDeviceFormat method 213
11.1.6 The getDeviceModel method 216
11.1.7 The getTerminalId method 217
11.2 Deploying a PvC adapter 218

11.2.1 Exporting the PvC adapter from VAJ to a JAR file 219
11.2.2 Deploying the PvC adapter JAR to the WCS server 219
11.2.3 Adding a PvC adapter definition to the WCS instance XML file . . 220
11.3 Deploying multiple PvC adapters 222
11.3.1 PvC adapter definition 222
11.3.2 Content management configuration 223

Chapter 12. Create, deploy and manage content 225
12.1 Content management configuration 226
12.1.1 Content management 226
12.1.2 Content management configuration 231
12.2 Create device-specific content JSPs 232
12.2.1 PvC data beans 232
12.2.2 Content development 233
12.2.3 Content examples of PvC features 234
12.3 Deploy content 238
12.4 Advanced content and configuration example 238
12.4.1 Auto selection of content 238
12.4.2 Reflection of terminal specification to JSP view 240
12.4.3 Content managed by adapter 242
12.4.4 Content managed by model 242

Chapter 13. Creating custom PvC commands 255
13.1 WCS V5.1 PvC command overview 256
13.1.1 PvC command summary 256
13.1.2 Scenarios for using the PvC commands 257
13.2 Create and deploy a custom PvC command 258
13.2.1 Create a custom PvC command 259
13.2.2 Deploy the custom PvC command 262

Chapter 14. HDML implementation sample 265
14.1 HDML toolkits and test clients 266
14.1.1 UP.SDK V3.2 for HDML 266
14.1.2 Mobile device hardware 269
14.2 Sample code for HDML 269
14.3 m-commerce direct development for HDML 270

Chapter 15. WAP implementation sample 271
15.1 WAP toolkits and test clients 272
15.1.1 Nokia WAP Toolkit V2.1 and simulator 272
15.1.2 UP.SDK for WAP 273
15.1.3 WAP mobile device hardware 273
15.2 Sample code for WAP 274
15.3 m-commerce direct development for WAP 275

Chapter 16. Palm implementation sample 279
16.1 Palm HTML browser implementation 280
16.1.1 Design guidelines 280
16.1.2 Development guidelines 282
16.1.3 Palm development tools 283
16.1.4 Sample code 284
16.2 Palm Web Clipping implementation 284
16.2.1 Design guidelines 284
16.2.2 Development guidelines 286
16.2.3 Testing 287
16.2.4 Problems 287
16.2.5 Example 288
16.2.6 Conclusion 288
16.3 Where to find more information 289

Chapter 17. i-mode implementation guidelines 291

Part 4. m-commerce using WTP implementation 293

Chapter 18. m-commerce using WTP implementation and design 295
18.1 Implementation considerations 296
18.1.1 WTP overview 296
18.1.2 Architecture for WCS-WTP integration 300
18.2 Application design guidelines using WTP 302
18.2.1 Introduction 302
18.2.2 Supported features 302
18.2.3 Usability 303
18.2.4 Trade-off 304

Chapter 19. m-commerce using WTP application development 305
19.1 Introduction 306
19.2 Sample code for WTP 306
19.3 HTML versus XML 307
19.3.1 HTML content using WTP 307
19.3.2 XML content using WTP 309
19.4 Selecting the right JSPs 315
19.5 Creating a generic PvC adapter 316
19.5.1 PvC adapter overview 316
19.5.2 Creating the PvC adapter 319
19.5.3 Deploying the PvC adapter 322
19.5.4 Content management 323
19.6 Creating the content JSPs 324
19.7 WCS caching with WTP 325

Part 5. Appendixes 327

Appendix A. PvC adapter framework reference 329
PvCAdapterImpl class and methods 330
PvC adapter definition for the WCS instance XML file 330
HTTP adapter attribute 332
PvC adapter attribute 333
IP adapter attribute 333
Excluded command 334
Protected command 334
Sample usage for PvC adapter definition for WCS instance XML file 334

Appendix B. PvC command reference 339
PVCRegistration 340
PVCRegistrationDevice 344
PVCChangeDevice 347
PVCBufferUrl 351
ReEnterPassword 353

Appendix C. PvC data bean reference 355
PVCBufferDataBean 356
UserPVCDeviceDataBean 356

Appendix D. PvC database tables and content management reference 359
PVCDEVMDL 360
PVCMDLSPEC 361
PVCDEVSPEC 363
PVCBinding 364
PVCSession 364
PVCBuffer 364
Content management reference 364
PVCDEVMLD table: define adapter default model 364
PVCDEVSPEC table: define minimum spec and JSP root 365
PVCMDLSPEC table: PVCDEVMDL and PVCDEVSPEC relationship . . 366

Appendix E. Additional material 369
Locating the Web material 369
Using the Web material 369
System requirements for downloading the Web material 370
How to use the Web material 370

Related publications 371
IBM Redbooks 371
Other resources 371

Referenced Web sites 372
How to get IBM Redbooks 373
IBM Redbooks collections 373

Special notices 375

Glossary 377

Index 383

Preface

This redbook provides developers and architects with the knowledge to develop and deploy m-commerce Web sites using WebSphere Commerce Suite V5.1.

Part 1, “Introduction to mobile commerce”, describes the concepts related to m-commerce, such as wireless technologies, development methodology for building m-commerce Web sites, and features in WebSphere Commerce Suite V5.1 for m-commerce. Also, we have included integration considerations regarding IBM wireless middleware and payment.

Part 2, “Setting up your m-commerce environment”, provides detailed instructions for setting up your runtime, development, and test environments. We provide a sample store and code to demonstrate IBM m-commerce functionality.

Part 3, “m-commerce direct implementation”, provides development guidelines for mobile devices directly accessing device-specific content JSPs. Samples and guidelines are included for WAP WML, HDML, Palm, and i-mode.

Part 4, “m-commerce using WTP implementation”, provides design and development guidelines for mobile devices using WTP to access WebSphere Commerce Suite HTML and XML content JSPs.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world at the IBM International Technical Support Organization, Raleigh Center.

Figure 0-1 The IBM Redbook team (1st row: John Ganci, Giuseppe Plagenza, Masaaki Ishibashi; 2nd row: Masaaki Saitoh, Peter Kovari, Rodrigo Magalhães, David Barker)

John Ganci is a Senior Software Engineer, WebSphere Specialist at the IBM ITSO Raleigh Center. He writes extensively and teaches IBM classes worldwide about all areas of WebSphere. John has 14 years of experience in product and application design, development, system testing, and consulting. His areas of expertise include e-commerce, personalization, pervasive computing, and Java programming. Before joining the ITSO, he developed e-commerce sites for IBM.

David Barker is an I/T Specialist in the IBM Software Group based in Hursley, U.K. He has two years of experience in the field of pervasive computing and three years of experience in Java programming. He holds a degree in Computer Science from Portsmouth University, U.K. His areas of expertise include pervasive computing, Java programming and XML.

Masaaki Ishibashi is a Software Developer at the Integrated System Evaluation Laboratory (ISEL), IBM Yamato Lab, in Japan. He has four years of experience in the e-commerce development field. He holds a Master's degree in Information Science from the Science University of Tokyo, Japan. He has written extensively about m-commerce application design and development; including i-mode and HDML. His areas of expertise include e-commerce software architecture and development, pervasive computing, and Java server programming.

Peter Kovari is a WebSphere Specialist at the IBM ITSO Raleigh Center. He writes extensively about all areas of WebSphere. His areas of expertise include e-business, e-commerce, and mobile computing. Before joining the ITSO, he worked as an I/T Specialist for IBM in Hungary.

Rodrigo Magalhães is a Consulting I/T Architect with the IBM e-business Competence Center in Rio de Janeiro, Brazil. He has seven years of experience in software development and consulting in the private sector, and has been designing and building Internet and intranet sites for five years. His areas of expertise include WebSphere Commerce Suite, pervasive computing, and Java server programming.

Giuseppe Plagenza is an Advisory I/T Specialist in IBM Global Services, Business Innovation Services, e-business Integration in Milan, Italy. He has several years of experience in the field of value-added services for mobile telecommunications networks. He holds a degree in Computer Science from the University of Pisa, Italy. His areas of expertise include SIM Application Toolkit, pervasive computing, wireless security and mobile commerce solutions.

Masaaki Saitoh is an Advisory Development Engineer in the Integrated System Evaluation Laboratory (ISEL), IBM Yamato Lab, Japan. He holds a degree in Fluid Dynamics from Waseda University, Tokyo. He has 13 years of experience in AIX system management and application development as an I/T Specialist in the Industrial Sector in Japan. He has written extensively about WebSphere Commerce Suite and i-mode.

Thanks to the following people for their contributions to this project:

Margaret Ticknor, IBM ITSO Raleigh Center, USA

Juan Rodriguez, IBM ITSO Raleigh Center, USA

Ivan Heninger, IBM Raleigh, USA

George Hall, IBM Raleigh, USA

Joanna Ng, IBM Toronto Lab, Canada

Mark Linehan, IBM Hawthorne, USA

Michal Jordan-Rozwadowski, IBM Toronto, Canada

Kazuhiro Yabuta, IBM Yamato Lab, Japan

Jakka Sairamesh, IBM Hawthorne, USA

Chung-Sheng Li, IBM Hawthorne, USA

Special notice

This publication is intended to help developers, I/T architects, I/T specialists, and consultants design, develop, test, and deploy m-commerce solutions. The information in this publication is not intended as the specification of any programming interfaces that are provided by WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000, or WebSphere Commerce Studio V5.1, Professional Developer's Edition for Windows NT and Windows 2000. See the PUBLICATIONS section of the IBM Programming Announcement for WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000, and WebSphere Commerce Studio V5.1, Professional Developer's Edition for Windows NT and Windows 2000 for more information about what publications are considered to be product documentation.

IBM trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

e (logo)® 
IBM ®
AIX
AS/400
DB2
DB2 Universal Database
Mobile Connect
MQSeries
Netfinity
OS/2
OS/390
PAL
PC 300
Redbooks
Redbooks Logo 
RS/6000
S/390
SecureWay
SP
SP1
ThinkPad
VisualAge
Wave
WebSphere
WorkPad

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to the address on page ii

Part 1

Introduction to mobile commerce

1

Introduction

In recent times, there has been a dramatic increase in the use of powerful mobile devices. These mobile devices, ranging from pagers to mobile phones, wireless PDAs, and wireless laptops, are changing the way people interact at work and at home. The sophistication of the mobile devices and wireless technologies has advanced to the stage of mass-market usage and acceptance. Just as we saw a surge in e-commerce by PC browser clients, we are now experiencing a similar phenomenon in mobile commerce by users of these more advanced mobile devices.

The combination of wireless technologies and the powerful Java-based application architecture provided by WebSphere Commerce Suite V5.1 promises to transform e-business. IBM is leading the way in the mobile computing arena in bringing together the technologies of the wireless world and the strength and integration of the IBM server and software family of products, to create end-to-end mobile commerce solutions.

Mobile computing, also known as pervasive computing, provides a series of technologies that enable people to accomplish personal and professional tasks using this new class of portable, intelligent, wireless mobile devices. These mobile devices give people access to information at any time and place. Some areas of the world are further invested in wireless technologies than others. However, the capability and number of users in all areas of the world are growing at a tremendous rate.

Wherever you live, mobile computing will in the very near future become a prominent means of accessing information on the Internet. Just as the PC browser client market matured from accessing the Internet simply for browsing and gathering information to full-blown e-business and e-commerce, the same is happening for mobile devices.

As a leader in e-business and e-commerce, IBM has recognized that mobile computing offers tremendous business opportunities, and has integrated mobile device support into its products and has developed a series of middleware products designed as infrastructure for mobile computing.

1.1 Mobile commerce overview

Before we start describing the features of WebSphere Commerce Suite V5.1 that provide mobile commerce support, it is important to understand some general concepts about m-commerce.

1.1.1 m-commerce defined

For the purposes of this redbook, the term m-commerce, or mobile commerce, refers to the use of mobile devices to partially or completely perform a transaction electronically from a commerce Web site for the exchange of goods or services for monetary consideration. Simply put, m-commerce is electronic commerce using a mobile device such as a mobile phone or PDA. This includes the following types of stores: Business-to-Consumer (B2C), Business-to-Business (B2B), and auctions.

The key distinction between m-commerce and e-commerce is the use of a mobile or wireless devices to access the commerce Web site, instead of using a wireline PC browser client.

We refer to mobile commerce throughout this redbook as m-commerce. The terms “mobile commerce”, “m-commerce”, and “wireless electronic commerce” are interchangeable.

1.1.2 Key objectives of m-commerce

We have listed the key objectives of the m-commerce initiative:

1. Enable WebSphere Commerce Suite V5.1 to support a wide range of mobile devices for m-commerce.
2. Provide an end-to-end solution for the mobile commerce customers by supplying wireless middleware products that integrate industry-standard wireless protocols with IBM servers and software products.
3. Create a flexible m-commerce product enablement architecture and infrastructure to keep pace with the fast-moving wireless technologies and standards.

1.1.3 Mobile devices

A mobile device, in the context of this redbook, is a portable, generally small wireless device that can be used to access the Internet via a browser. All mobile devices are not created equal. There is a wide range of capabilities and functionality. In addition, the capability between the mobile device categories continues to be blurred. For example, some phone manufacturers are developing PDA-like functionality into mobile phones and providing larger screens and easier methods of input.

We have categorized the mobile devices as follows:

- Mobile phones
- Wireless PDAs
- Wireless laptops

Mobile phones

When we refer to a mobile phone in this redbook, we are talking about mobile or wireless phones that have a microbrowser to access Internet content. Other names for a mobile phone include “cell phone”, and “wireless phone”. A standard mobile phone includes the following capabilities:

- Voice
- Messaging (SMS or WAP push)
- Data (access to Internet content via a microbrowser)

For the purposes of this book, we have excluded analog cell or mobile phones that only provide voice capability. There are two major protocols that are used by mobile phones capable of Internet browsing: WAP and i-mode.

Wireless PDAs

A Personal Digital Assistant (PDA) is a handheld computer with a wireless interface that serves as an organizer for personal information. PDAs often have a pen-like stylus to tap selections on menus and to enter printed characters. The unit may also include a small on-screen keyboard that is tapped with the pen. Data is often synchronized between the PDA and desktop computer via cable or wireless transmission.

We are interested in PDAs that have wireless transmission capability and include a Web browser. Some examples of PDAs are the Palm VIIx, Epoc, and Pocket PC.

Wireless laptops

This category includes laptops, notebooks, or portable PC browser clients that have a wireless interface to the network for Internet access. These clients use standard TCP/IP protocols and a standard browser, such as Netscape Navigator or Microsoft Internet Explorer. The wireless connection is usually much slower than for wireline-based network clients. An example of a wireless notebook is the IBM ThinkPad T20 with a wireless PC card adapter.

1.1.4 Challenges

We are now in a period where the mobile technology offers capabilities that make it useful and attractive for business and consumers to invest in mobile commerce. As the functionality of the mobile devices continues to improve, the challenge will be for the technology infrastructure providers and application developers to keep pace with the new devices, networks, protocols and standards.

1.2 m-commerce opportunities and market drivers

In this section, we will examine the business opportunities and key market drivers for the existing and future growth of the mobile and m-commerce market:

- m-commerce opportunities
- Mobile market penetration
- Technology advances
- New business models
- Customer needs

1.2.1 m-commerce opportunities

The mobile Internet application market is exploding with real revenue opportunities. By 2003, data enabled mobile phones will overtake the number of PCs in use. IBM anticipates that the m-commerce market will reach $100 billion in 2003. Figure 1-1 displays some staggering estimates of the number of mobile devices in use now and in the future, as well as the projected I/T revenues.

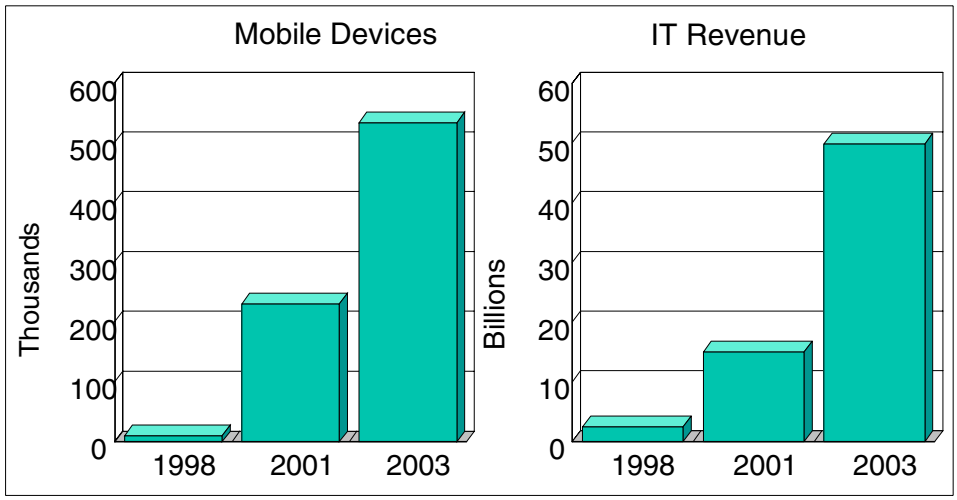


Figure 1-1 m-commerce opportunities

1.2.2 Mobile market penetration

The mobile communications market has reached the mass market in certain areas and is growing rapidly in others. We have selected three areas that demonstrate the penetration in the worldwide market:

- Mobile computing in Japan
- Mobile computing in Europe
- Mobile computing in the United States

Mobile computing in Japan

There are two primary types of mobile devices used in Japan: the mobile phone and the Personal HandyPhone System (PHS). The PHS is a kind of cell phone with limited functionality compared to the mobile phone, and is not considered for the purposes of this redbook.

Carriers

There are three major carriers for wireless connectivity of mobile phones in Japan:

- NTT DoCoMo
- KDDI
- J-PHONE

Each carrier providing service for mobile phones has provided Internet connectivity service for their own mobile phone users since 1999. As of February 2001, the number of mobile phone Internet users has reached 30 million. Considering that the Japanese population is approximately 120 million, almost one in four people can connect to the Internet by mobile phone. Table 1-1 provides a summary of the number of Internet users by carrier and service in Japan.

Table 1-1 Number of mobile Internet users by carrier in Japan

Service Name		**i-mode**	**EZweb**	**J-SKY**	**Total**
Carrier Name		**NTT DoCoMo**	**KDDI**	**J-PHONE**	
Number of mobile phone users (1)		35,115,000	1,462,000	9,726,000	59,456,000
Number of mobile phone internet users (1)		19,777,000	6,114,000	5,521,000	31,411,000
Number of Web sites for mobile phone	Official	1,600 (2)	680 (3)	550 (4)	N/A
	Voluntary	38,000	N/A	N/A	N/A

Note: Figures obtained February 2001 from the following sources:

(1) http://www.tca.or.jp/index-e.html (current figures)
(2) http://www.nttdocomo.co.jp/i/
(3) http://www.au.kddi.com/ezweb/contents/index.html
(4) http://www.j-sky.j-phone.com/

The number of mobile Internet users has increased rapidly in Japan as seen in Figure 1-2 on page 10.

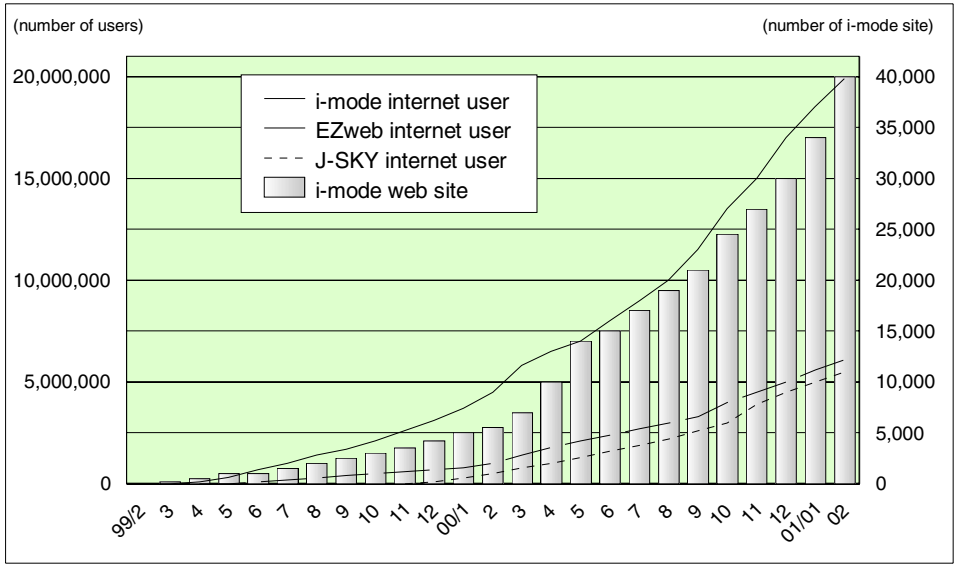


Figure 1-2 Mobile Internet user growth in Japan

The content providers are starting to support mobile phones as well as PC browsers for nearly all content on Web sites. For example, content supported for mobile devices includes news, weather forecast, banking, credit cards, stock trading, transportation, airline reservations, hotel reservations, map services, online tickets for trains or concerts, books, music CDs, videos, games, used cars, real estates, fashion, health, pets, restaurants, sports, entertainment, lottery, TV timetable, job search, government services, portal sites, e-mail services, and so on.

Note: The dominant wireless service provider in Japan is NTT DoCoMo, which uses i-mode. For more information on i-mode, refer to 2.2.3, “i-mode” on page 46.

Mobile computing in Europe

In the European mobile market, WAP is the dominant protocol used for mobile phones. Figure 1-3 provides a good summary of the current number of mobile-phone users in Europe by country, as well as the expected growth through 2003.

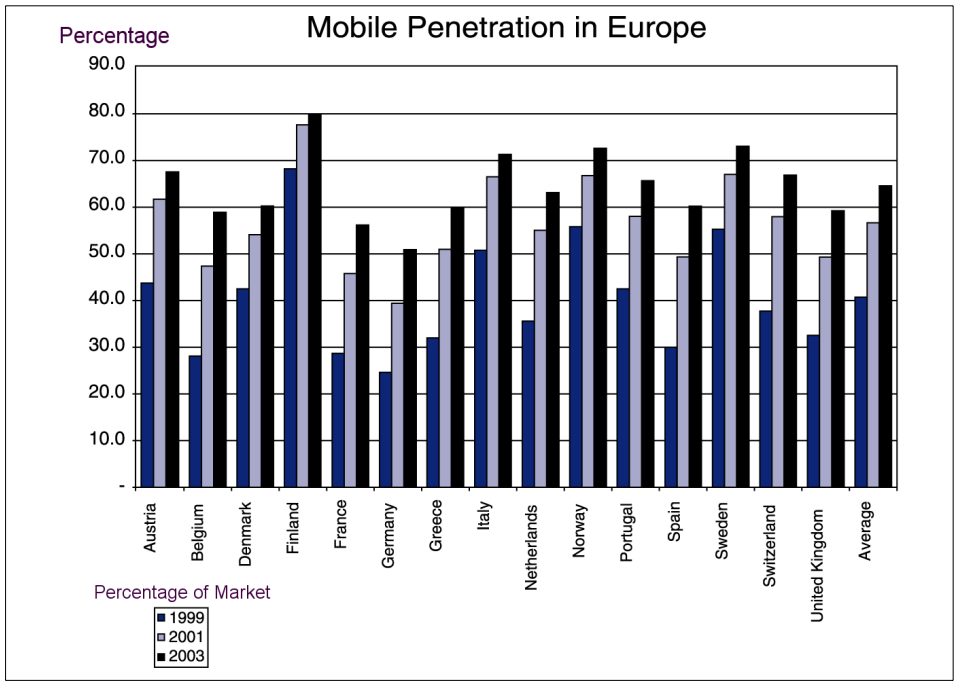


Figure 1-3 Mobile penetration in Europe

Mobile computing in the US

In the US market, there is no one dominant technology used for mobile devices. The Palm VII wireless PDA has been successful by providing a Web-clipping solution to access existing HTML-based sites. The Pocket PC PDA, which uses the Microsoft Windows CE operating system, is gaining momentum. The new Microsoft initiative code named “Stinger” offers promising capability at the browser level by providing markup language support for HTML, WML (WAP), and cHTML (i-mode).

Technology providers from Japan and Europe are attempting to get a foothold in the US market by signing agreements with US service providers. For example, NTT DoCoMo, based in Japan, has recently signed an agreement with AT&T to provide i-mode mobile phone service in the US. Carriers such as Cingular Wireless, which provides WAP mobile phone support, have signed service agreements with European carriers offering the same type of technology.

1.2.3 Technology

Technology is driving the mobile market from multiple directions. Two very significant market drivers of technology are technological advances and the technology that suppliers push.

Many of the technology limitations for mass market appeal have been overcome. It is now more a question of service providers and wireless infrastructure providers being able to implement these new mobile technologies. Once these new technologies are in service, the application developers will need to incorporate the new technologies into m-commerce applications. The technology suppliers and service providers are eager to cash in on the huge investments made in creating the wireless technologies.

Faster wireless networks, more powerful Internet-capable mobile devices, and applications such as m-commerce are all factors driving the market.

1.2.4 New business models

Many new business models have been established recently that include the use of mobile devices. The mobile device has its special characteristics: portability, low cost, more personalization, GPS, voice and so on. In this section, we describe the new business models evolving to leverage the unique functionality and opportunities mobile devices offer.

Content distribution services

Users want to stay connected and are interested in the latest information. Some examples of content distribution services are as follows:

- Real-time information

  Real-time information such as news, stock prices or tonight's weather forecast can be distributed to your personal mobile phone via the Internet during the morning commute time. Content providers can send the information that is personalized to the user's interest. For example, during the Sydney Olympic Games, a service was offered that sent e-mail automatically to a mobile phone when the subscriber’s home country won a medal.

- Notification

  Notices can be sent to the mobile device; for example, a stock price has hit a user-defined threshold, or the book that was reserved has just arrived at the store, or the result of the professional baseball game, and so on.

- Positioning system to personalize information is known by location

  By using a positioning system, you can retrieve local information. For example, while you are in a shopping mall, a supermarket can advertise a temporary discount service directly to your mobile device. Similarly you can get traffic information along the road you are driving.

- Advertising

  Mobile devices offer the advertising market a higher rate of response than can be expected on a PC. Mobile device terminals are generally small, so an advertisement may occupy a relatively large portion of the screen and catch user's eye more easily.

Online financial service

The online financial service industry has been a leader in deploying mobile solutions. A balance inquiry of an account or payment treatment can be done while you are drinking a cup of coffee at lunchtime. Or you can buy some stocks by checking the graphical stock growth chart on the mobile phone display.

In Japan, there are over 450 banks, eight securities, eight life insurance companies, and 10 credit card companies already in service using i-mode, provided by the NTT DoCoMo official site.

Reference service

The mobile Internet is a very powerful tool for accessing basic reference information. In combination with a positioning system, a Web site can locate a suitable restaurant in the neighborhood, display the route to that restaurant on a map, make a reservation, and show you a menu of today's specials.

Online shopping

Generally speaking, the quality of the image files displayed on a mobile device screen is relatively poor. On the other hand, items such as music CDs, games, books, and tickets are good examples of products that do not need high image quality for display on mobile devices.

Repeat orders can be easily issued by referring to a previously purchased item from the commerce site.

Auctions are also easily accessible via mobile devices. For example, your m-commerce application can notify you of an increase or decline of your bid.

The mobile device can be used to identify the user accessing the Web site. Registered users do not need to enter their personal information such as credit card number or phone number. In the near future, the mobile device itself will act as a personal identification device and be used as an electronic wallet.

Corporate intranet

Mobile computing provides a number of benefits for any kind of business if a mobile device is connected to the office information system. For example, your notebook PC can access your company's intranet from outside of the office, without a phone line.

Some examples of corporate intranet use of mobile devices are:

- Real estate salesperson

  If you are a real estate salesperson, you can get information about land or homes for sale, with the assistance of the mobile device.

- Delivery driver

  If you are a delivery driver, your geographical position and the transaction you performed can be reported to the headquarters by entering the delivery record using the mobile phone. This information can be used to determine your position with a GPS in your phone and then find the best route for your next delivery.

1.2.5 Customer needs

As the capability of mobile devices improves, users will be able to access information anytime and anywhere. Consumers that currently use their mobile phones primarily for voice and SMS messages are now able to use a mobile phone to engage in m-commerce as well as browse the Internet. Sales, services, and logistics-related businesses have great potential to benefit from the functionality provided by m-commerce.

1.3 IBM m-commerce

IBM provides a wide range of electronic commerce solutions for your business needs. Within the IBM WebSphere brand, IBM has enabled two electronic commerce product suites to support m-commerce:

- IBM WebSphere Commerce Suite V5.1

  This product suite is very versatile and can be used for B2C, B2B, and Auction e-commerce Web sites. WebSphere Commerce Suite V5.1 provides a pure Java programming model that is conducive to supporting mobile clients. Features have been added to WCS V5.1 to provide integration and support for mobile devices.

- IBM WebSphere Commerce, MarketPlace Edition V4.2

  This product suite is designed specifically for electronic marketplaces. Version 4.2 of this product provides extensions for m-commerce.

Refer to the following URL for more information on this product:

`http://www.ibm.com/software/webservers/commerce/wcs_me/index.html`

1.3.1 m-commerce using WebSphere Commerce Suite V5.1

WebSphere Commerce Suite V5.1 allows customers to easily build m-commerce Web sites for a wide range of mobile devices. The m-commerce solutions are built using the WebSphere Commerce Studio, and deployed in the runtime environment using WebSphere Commerce Suite.

WebSphere Commerce Studio

WebSphere Commerce Studio includes two major products that are used to develop m-commerce or e-commerce Web sites: WebSphere Studio V3.5 and VisualAge for Java V3.5, Enterprise Edition.

We provide detailed information on the m-commerce development environment in Chapter 8, “m-commerce development environment” on page 159.

WebSphere Commerce Suite

WebSphere Commerce Suite V5.1 includes the following major products used in the runtime environment for m-commerce and e-commerce Web sites:

- IBM HTTP Server V1.3.6.12
- IBM DB2 V7.1 Enterprise Edition
- IBM WebSphere Application Server, V3.5 Advanced Edition
- IBM WebSphere Commerce Suite V5.1
- IBM WebSphere Payment Manager V2.2

We provide detailed information on the m-commerce runtime environment in Chapter 7, “m-commerce runtime environment” on page 141.

m-commerce enablement features in WCS V5.1

WebSphere Commerce Suite V5.1 includes PvC (pervasive computing) extensions designed to enable m-commerce. The PvC extensions provide support for session control, device control, security enhancement, and URL buffering.

Below is a summary of the features added to support m-commerce in WebSphere Commerce Suite V5.1:

- PvC adapter framework

  The PvC adapter framework provides Java classes and methods to create a PvC adapter for many wireless protocols. The PvC adapter provides support for session control, detects of the device type, checks the device format for subscriber ID, and sets the root path of the content JSPs for the specific or category of mobile device.

  - For a detailed description of the PvC adapter framework, refer to 4.2.1, “PvC adapter framework” on page 87.
  - For detailed instructions on creating and deploying a PvC adapter, refer to Chapter 11, “Creating and deploying a PvC adapter” on page 205.
  - For reference information on the syntax and usage of the PvC adapter framework, refer to Appendix A, “PvC adapter framework reference” on page 329.

- PvC commands

  The PvC commands are used to develop custom WCS commands written in Java. The commands are used within the content JSPs to solve specific mobile commerce issues such as device security and URL buffering.

  - For a detailed description of the PvC commands, refer to 4.2.2, “PvC commands” on page 90.
  - For detailed instructions on creating custom PvC commands, refer to Chapter 13, “Creating custom PvC commands” on page 255.
  - For information on syntax and a usage example, refer to Appendix B, “PvC command reference” on page 339.

- PvC data beans

  The PvC data beans are used to access information related to the PvC commands and database tables for information about the detected mobile device type.

  - For a detailed description of the PvC data beans, refer to 4.2.3, “PvC data beans” on page 91.
  - For information on syntax and a usage example, refer to Appendix C, “PvC data bean reference” on page 355.

1.4 Structure of this redbook

This redbook is organized in four parts and designed to provide information for different audiences such as architects, developers, sales professionals, and I/T specialists.

Part 1: Introduction to mobile commerce

Part 1 of this redbook is intended for sales and marketing professionals as well as developers, architects, and specialists. The chapters in this part of the redbook provides a foundation for m-commerce development and deployment by exploring the mobile commerce concepts and features, wireless technologies, and integration considerations for implementing an m-commerce solution.

The chapters in Part 1 of this redbook are organized as follows:

- Chapter 1, “Introduction”, provides an overview of m-commerce, describes the opportunities and market drivers for m-commerce, and explains the features added to WCS V5.1 to enable m-commerce.
- Chapter 2, “Wireless technologies”, provides an overview of the wireless technologies that are used to enable m-commerce.
- Chapter 3, “m-commerce development methodology”, guides developers and architects through the process of designing and developing m-commerce Web sites using WCS V5.1
- Chapter 4, “m-commerce features and functionality in WCS V5.1”, provides a description of the features and functionality added to WCS V5.1 to enable m-commerce.
- Chapter 5, “IBM wireless middleware”, describes the m-commerce integration considerations when using IBM wireless middleware.
- Chapter 6, “m-commerce payment solutions”, explores m-commerce payment solutions used within the industry and solutions available using WebSphere Payment Manager.

Part 2: Setting up the m-commerce environment

Part 2 of this redbook is intended for developers and architects and provides a hands-on approach to setting up the m-commerce development, runtime, and test environment. In addition, we provide a sample e-commerce Web site to use as a base upon which an m-commerce Web site can be built, as well as sample code that will be used throughout the redbook.

The chapters in Part 2 of this redbook are organized as follows:

- Chapter 7, “m-commerce runtime environment”, provides an overview of the production runtime environment, as well as instructions for implementing a runtime environment for the purposes of testing an m-commerce Web site.
- Chapter 8, “m-commerce development environment”, provides instructions for setting up the m-commerce development and test environments.
- Chapter 9, “m-commerce sample store and sample code”, provides instructions for deploying a sample store to enable m-commerce, along with sample code to be used in the implementation process.

Part 3: m-commerce direct implementation

Part 3 of the redbook is intended for developers and architects, and provides development guidelines for mobile devices directly accessing device-specific content JSPs. Samples and guidelines are included for WAP WML, HDML, Palm, and i-mode.

The chapters in Part 3 of this redbook are organized as follows:

- Chapter 10, “m-commerce direct design and development process” on page 199, provides design and development guidelines for developing a store using the direct m-commerce approach.
- Chapter 11, “Creating and deploying a PvC adapter” on page 205, explains in detail how to create and deploy a PvC adapter.
- Chapter 12, “Create, deploy and manage content” on page 225, provides instructions for content management configuration, content development, and content deployment. In addition, we include an advanced content development and configuration example.
- Chapter 13, “Creating custom PvC commands” on page 255, describes how to create a custom PvC command.
- Chapter 14, “HDML implementation sample” on page 265, provides guidelines and sample code for developing a store with HDML content.
- Chapter 15, “WAP implementation sample” on page 271, provides guidelines and sample code on developing a store with WAP WML content.
- Chapter 16, “Palm implementation sample” on page 279, provides guidelines and sample code on developing a store with content for Palm devices using HTML or Web Clipping.

Part 4: m-commerce using WTP implementation

This part of the book is intended for developers and architects and provides design and development guidelines for mobile devices using WTP to access the WCS HTML or XML content JSPs.

2

Wireless technologies

This chapter provides readers not familiar with wireless technologies a foundation for understanding the unique issues faced by project managers, architects, developers and specialists when developing and deploying mobile commerce applications. Many of the chapters to follow in this redbook make references to various wireless networks, protocols, mobile devices, and markup languages discussed in this chapter.

The chapter is organized into the following sections:

- Wireless networks
- Wireless protocols
- Mobile devices
- Content and markup languages
- Wireless service providers
- Next-generation technologies

2.1 Wireless networks

Wireless networks are used to transmit data between mobile devices or personal computers using wireless adapters without the use of a physical cable or wire. In this section, we will describe the wireless networks in use today, as well as the wireless networks of the future.

This section is organized into the following topics:

- Mobile communications network history
- GSM
- GPRS
- Mobitex
- CDPD
- IMT-2000
- Wireless LANS

2.1.1 Mobile communications network history

The birth of wireless mobile communications can be traced back to the invention of frequency modulation (FM) in 1935 by E.H. Armstrong. The first wireless mobile network was introduced in the United States in 1946, when the Federal Communications Commission (FCC) granted AT&T the first mobile network license to cover the area of St. Louis. The system was based on frequency modulation and the area coverage was guaranteed by a unique antenna, installed at the highest point in the city. The user needed to manually scan for an available channel (manual trunking) and then communicate over this channel with an operator to set up the call. Such a system was half-duplex, since it allowed only one person to talk at a time.

A big improvement was introduced in 1962, with the Improved Mobile Telephone Service (IMTS) network. This system exploited more channels, in order to handle a higher number of simultaneous users. Moreover, the call procedure was made automatic, with an automatic trunking of the available channels, which eliminated the need for operators. However, in spite of the capacity improvements achieved, such systems still presented enormous limitations in terms of the maximum number of users that could be handled simultaneously.

The turning-point came with cellular systems. The first of such networks was the Advanced Mobile Phone System (AMPS), introduced in 1983 in the United States. Cellular networks are based on the technique of frequency reuse, which optimizes the use of the limited radio spectrum.

In cellular radio networks, the area covered by one base station (for example, an antenna) is reduced and other base stations are installed with small overlapping areas. Adjacent cells (coverage area) need to use different frequencies to avoid interferences, but the same frequency can be reused in non-adjacent cells, as seen in Figure 2-1.

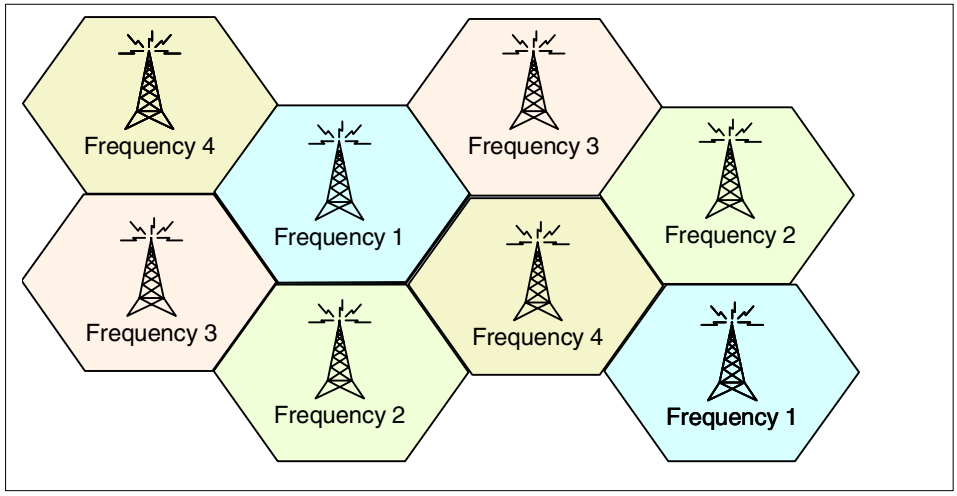


Figure 2-1 Frequency reuse

Due to the splitting of the coverage area into many cells, it may happen that a moving user passes from one cell to another. In this case, in order to guarantee the highest communication quality, the mobile device needs to switch to one of the frequencies of the new cell. Since every antenna uses a limited number of frequencies, the device will select the one that offers the best quality. However, this change of frequency also needs to be transparent for the user and should not affect any call in progress. The procedure by which the terminal changes the frequency when passing from one cell to another is called handover.

By splitting an area into many smaller cells, the overall network capacity increases. However, the cell size cannot be decreased beyond a certain threshold. In fact, a decrease in the cell size, means a decrease in the reuse of frequency. Consequently, the probability of interferences and handovers also increases.

Determining the optimal size of each cell is a fundamental issue in the radio coverage planning of a certain area. The major factors to consider are:

- Presence of reliefs
- User density
- Area topology (urban, suburban, rural)
- Presence of buildings

When a mobile device is switched on, it will scan a predetermined set of radio channels to find a base station. After finding one, the terminal tries to get access to the network by means of a registration procedure, in which the device identifies itself to the network with a unique identifier. If the registration succeeds, the terminal can access the network and from that moment onward the location of the device within the coverage area is kept under control, in order for the network to direct any incoming calls to the proper cell.

It may happen that a mobile device is in an area that is covered by a network provider other than its own. In that case, if the two networks are based on the same technology and an agreement between the two involved operators exists, the terminal can access its home network through the infrastructure of this hosting operator. For this purpose, the hosting network will communicate with the home network to check whether access can be granted. This kind of agreement is called a roaming agreement and covers interpretability as well as billing issues.

Networks are either circuit-switched or packet-switched. With circuit-switched networks, the network resources are assigned to a single connection for the whole duration of the communication. This dedicated channel remains allocated even in those time frames when no data is sent over the channel. Voice traffic is typically sent over circuit-switched connections. The GSM network also supports data traffic over circuit-switched connections. However, data services are very different in nature from voice services. In typical voice communications, you usually have very short idle times between two consecutive transmissions, which means that the allocated channel is intensively used during the communication. The situation for data traffic is quite different. Consider the case of a user browsing the Internet. When a user sends an HTTP request from a PC browser client, they then receive a response page for display of the content. In this phase, the user is operating offline and is not using the network. At that time, the resource could be assigned to other users waiting to transmit. This is the typical situation of data services and in such cases a packet-switched technology is highly preferable.

In packet-switched networks, no permanent channel is allocated to a single connection. Data is transmitted in the form of packets and the network resources are allocated on-demand. This also means that any single channel can be shared among different users and that the network resources are better exploited, because they are allocated only when effectively needed.

Adopting a packet-switched network for data traffic also allows for more flexible and convenient billing plans. In a circuit-switched connection, the user has to pay for the whole duration of the connection, and then even for those time frames when the network is not used. With a packet-based network, the user pays for

service based on the amount of data packets downloaded. Moreover, with this kind of network, flat rates are applicable. For example, the user could pay a fixed amount of money a month, provided the download traffic in that month has not exceeded a certain threshold.

Wireless network generations

When looking at how wireless networks have evolved, the different phases and characteristics of wireless networks are often referred to as generations: 1G for first generation, 2G for second generation, etc.

The first generation (1G) cellular systems were analog and implemented with such technologies as AMPS. The AMPS 1G network was introduced in 1983 in the US and was used for voice capability.

Second-generation (2G) systems were introduced in the 1990s. Some of the technologies implemented include TDMA, CDMA, and GSM. 2G systems were used primarily for voice, with the exception of SMS offered by GSM.

Some 2.5G systems have been implemented recently, such as HSCSD and GPRS. HSCSD is a circuit-switched extension of GSM that allows an increase in the bit rate by combining more than one timeslot in the GSM radio interface. GPRS can be considered a packet-based extension of GSM and provides higher data throughput. Other 2.5G systems, such as EDGE, are still to be implemented.

Third-generation (3G) systems are expected to be implemented over the next few years and will be called UMTS in Europe and IMT-2000 worldwide. At the time of writing this redbook, NTT DoCoMo is in the process of launching a 3G IMT-2000 cdmaOne-based wireless network in Japan. 3G networks provide higher-speed transmission to support high-quality audio and video, as well as global roaming capability.

Table 2-1 provides a summary of the wireless networks in use today, and a look at future implementations.

Table 2-1 Implementation of wireless networks

Wireless network type	Currently in use	Near-term implementation	Long-term implementation
GSM	Europe (900, 1800) US (1900)	US (1900)	
GPRS	Europe	Europe	
Mobitex	Europe, US		
CDPD	US		

Wireless network type	Currently in use	Near-term implementation	Long-term implementation
PDC	Japan	Japan	
cdmaOne	US, Korea, Japan	US, Korea, Japan	
IMT-2000		Japan	Japan, US
UMTS			Europe

2.1.2 GSM

The Global Systems for Mobile Communications (GSM) standard was proposed by CEPT in 1982 and completed by ETSI in 1990, while the first networks were deployed in 1991. The main reason behind the introduction of GSM in Europe was to provide a common standard for European Cellular Communications, which allowed subscribers to roam throughout Europe and access cellular networks in each country with the same equipment. It also allowed the equipment manufacturers to sell identical mobile and fixed equipment in all countries. GSM technology has been evolving through three different phases of the standard specifications, as seen in Table 2-2.

Table 2-2 GSM phases of standard specifications

GSM phase	Description
Phase 1	This was the first specification, completed in March 1995.
Phase 2	This version was completed at the end of 1995 and added some new features to the previous release, such as supplementary services (for example, call waiting) and enhanced full rate.
Phase 2+	This is the latest version of the GSM specification and introduces interesting concepts such as HSCSD, GPRS, EDGE and the SIM application toolkit.

GSM base stations transmit toward the terminals (downlink) on channels in the range of frequencies 935-960 MHz (25 MHz range), and receive in the range 890-915 MHz (25 MHz range). The transmit and receive frequencies for any channel are always separated by 45 MHz. The GSM technology supports both voice and data traffic over circuit-switched connections and provides a data throughput of 9.6 kbps. The radio interface uses both Time Division Multiple Access (TDMA) and Frequency Division Multiple Access (FDMA). The available range of frequencies is divided into many GSM carriers (FDMA) and each GSM carrier is divided into groups of eight timeslots (TDMA). In GSM no user can

transmit over more than one timeslot at the same time. Moreover, during a connection each user is allocated one timeslot to transmit and one timeslot to receive. These situations are different in other technologies, such as GPRS (see 2.1.3, “GPRS” on page 32).

Today, GSM is one of the most important and widespread mobile standards worldwide. Many variants of GSM have been created for different frequency ranges. For instance, ETSI developed the Digital Communications System 1800 standard (DCS1800) as an extension to GSM. It operates in the 1800 MHz band in Europe, uses less power for mobile phones, and has smaller cell sizes. GSM and DCS1800 are functionally similar in terms of voice quality, data facilities, and call handling, and the user can hardly note the difference. GSM technology in the form of DCS1900 is available in North America, and often referred to as Personal Communications Services (PCS) systems.

In a GSM network, the subscriber is considered an entity separate from the device. This means that the subscriber identity can be transferred from one physical phone to another, without reprogramming the device. This is accomplished by means of a Subscriber Identity Module (SIM), which is a small smartcard to insert into the mobile phone. The first SIM cards were credit card-sized but now nearly all GSM phones support only the so-called plug-in format, which corresponds to the break-out section shown in Figure 2-2 (opposite side shown in Figure 2-3).

Figure 2-2 GSM Subscriber Identity Module (SIM)

Figure 2-3 SIM other side

The SIM can be inserted into a GSM phone, as seen in Figure 2-4, which then takes the identity of the subscriber's GSM phone.

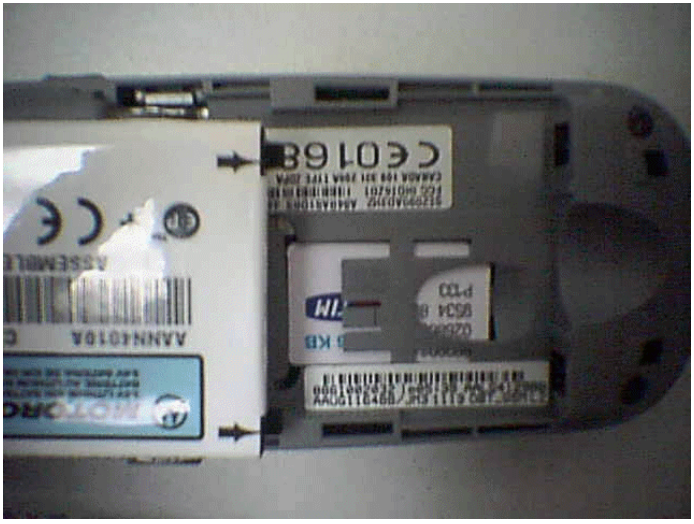


Figure 2-4 SIM inserted into a WAP phone

The SIM card technology has been evolving very quickly in the last two years. In spite of their very reduced size, SIM cards come with a CPU, EEPROM memory for the file system, a ROM memory for the card operating system, and a RAM memory for program execution. The first SIM cards were only used to store data and the secret keys needed by the GSM security algorithms for authentication and encryption.

The CPU processing power and the available memory allow Java applications to run on the card itself. Some operators in Europe have already launched many services based on the combination of Java SIM cards and the SIM Application Toolkit, which is a standardized technology to develop applications on the SIM cards.

Each GSM subscriber is uniquely identified by the International Mobile Subscriber Identity (IMSI), which is stored on the SIM card, can have a maximum length of 15 bytes and contains the following pieces of information:

- Mobile Country Code (3 digits) - Country code of the operator
- Mobile Network Code (2 digits) - Operator code within the country
- Mobile Station Identification Number (max 13 digits) - Serial number

The physical device is identified by the International Mobile Equipment Identifier (IMEI), that is a 15 byte-long code embedded in the device.

One interesting service of the GSM network is the Short Message Service Point-to-Point (SMS-PP). It allows the GSM phone to send and receive short messages, using the network's control channel to transfer the information. This means that no circuit-switched connection is established and the message can be stored, then forwarded when the phone is able to accept the message. The GSM Short Message Service allows the phone to act as a two-way alphanumeric pager. Up to 160 characters can be transmitted in one message. The received messages are usually stored on the SIM card. Each message has a validity period parameter, which can be set up on the phone and states the overall lifetime of the message. If the message has not been delivered within the validity period, it will be discarded by the network. To guarantee this service, one or more Short Message Service Center (SMSC) nodes need to be attached to the GSM signalling network. SMS messages are currently the most widespread means of sending data over GSM networks. According to the GSM Association, a record 15 billion SMS messages were sent over the world's GSM wireless networks during December 2000.

Other interesting data services of the GSM technology are the Short Message Service Cell Broadcast (SMS-CB) and the Unstructured Supplementary Services Data (USSD).

The SMS-CB allows the user to send messages from the network to many different mobile phones simultaneously. This is achieved by sending the message to a base transmission station (BTS) with an indication of all the cells depending on that base station controller (BSC) where the message should be broadcasted. Each message can be 1 to 15 pages long. One page can contain 93 alphanumeric characters or 82 binary octets in length. Many messages can also be concatenated.

Note: This is important because, contrary to the SMS we are familiar with (called SMS-PP), SMS-CB does not allow the user to send messages but only to receive them. Only the network has the capability to broadcast information.

The USSD is another means of sending messages over a GSM network. It is different from SMS because it is not a store-and-forward service and because it provides session management. A new session is established when the user first accesses the USSD service. The message length is 182 bytes.

The GSM architecture is very complex, as seen in Figure 2-5.

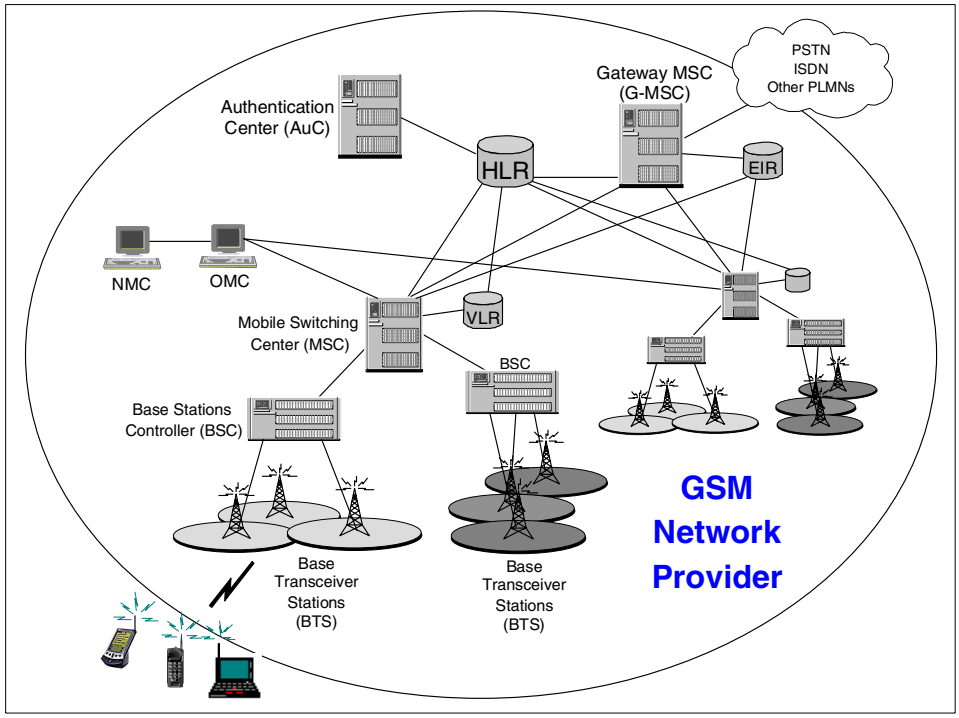


Figure 2-5 GSM network architecture

Each of the components has a well-defined role within the overall architecture:

- **Base Transceiver Station (BTS)**

  BTSs in GSM perform tasks similar to the base stations of other networks. Each BTS can be defined as the set of all transceivers (for example, transmitters and receivers) and equipment that can provide radio coverage within a cell.

- **Base Station Controller (BSC)**

  The BSC controls several base stations, manages the settings of the radio channels and ensures that a mobile phone can move from one cell to another and switch to the new radio channel without a communication break. The BSC can only manage internal handovers (for example, between cells under its own control).

- **Mobile Switching Center (MSC)**

  MSCs are the main switching nodes within GSM networks. In one network there are usually many MSCs, each one controlling many BSCs and therefore a huge area. The main functions of the MSC are call setup, call control, mobility management and generation of billing data. It is essentially in charge of properly routing any incoming or outgoing calls for all the terminals that are in its coverage area. Finally, it can also manage all the handovers between cells belonging to different BSCs. Some MSCs also handle special functions for the SMS service.

- **Visitor Location Register (VLR)**

  The VLR is a database that contains all the information related to the users that are within the coverage area of its own MSC. This data is needed for call control and management. As soon as a terminal enters the area covered by a new MSC, the VLR queries the HLR to get the needed user data. This user is then registered in the VLR of this new MSC and a reference to this VLR is associated to the same user in the HLR. Each MSC and the corresponding VLR are often deployed as a single entity (for example, they are integrated in a single node).

- **Home Location Register (HLR)**

  The HLR is the main node of the GSM network. It is basically a huge database used to store permanently the data related to all subscriptions, such as service profile and activity status.

- **Gateway MSC (G-MSC)**

  The G-MSC is a special MSC that interfaces the GSM network with either a fixed phone network (PSTN or ISDN) or the mobile network of a different operator (PLMN). PLMN is the acronym for Public Land Mobile Network.

- **Equipment Identity Register (EIR)**

  The EIR is a database of device identifiers (IMEI). It contains devices that are allowed to access the network, devices that cannot access the network, and devices that need to be located (for example, this database can be used to detect the users of stolen mobile phones and locate them).

- **Authentication Center (AuC)**

  The AuC is in charge of computing the security parameters to authenticate the user and encrypt the traffic.

- **Operations Management Center (OMC)**

  The OMC manages the network on a regional and day-to-day basis. It is used to configure the various nodes of the network, to monitor faults and to collect accounting data.

- **Network Management Center (NMC)**

  The NMC manages the entire network on a global basis and is used for long-term planning. Usually, there are many OMCs, and only one NMC in a network.

Regarding security, GSM networks provide authentication and confidentiality. Authentication is mutual, in the sense that the terminal and the network authenticate each other, and it takes place when any attempt of the mobile device to access the network is made. This means that authentication is also needed when the user wants to place a call. GSM authentication is based on a challenge-response protocol and is performed via the AUC and the SIM card. The GSM authentication protocol is depicted in Figure 2-6.

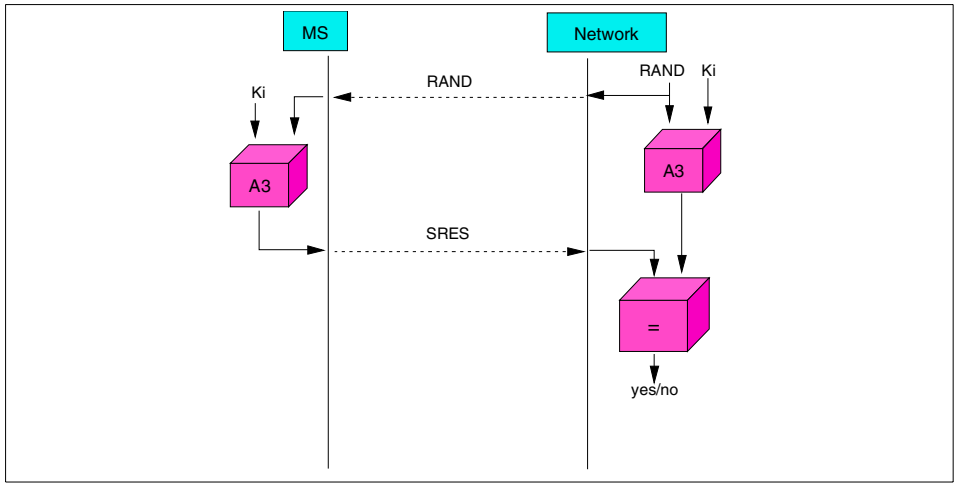


Figure 2-6 GSM authentication protocol

The major elements involved in the authentication process are:

- The authentication key *Ki*
- The *A3* authentication algorithm

The A3 algorithm is the same for all GSM users, whereas each subscriber has his/her own authentication key stored both on the SIM card and in the HLR. The overall authentication procedure is the following:

1. The network sends a random value (RAND) to the mobile device.
2. The terminal uses the received value and the stored Ki as input values for the A3 algorithm, and sends back the result (SRES) to the network.
3. The network compares the received value with the one it has computed itself. If the two values are equal, the authentication is successful; otherwise it fails.

The computation of the random value and the result SRES on the network side is the responsibility of the AuC.

Similar considerations apply for the generation of the encryption key. Exactly like the authentication procedure, this process is performed many times in order to have a different ciphering key for any communication. The authentication key Ki is also involved in this second procedure. The algorithm that is used to generate the key is the A8 algorithm. Both the network and terminal compute the same encryption key Kc by applying the A8 algorithm to the Ki, as seen in Figure 2-7.

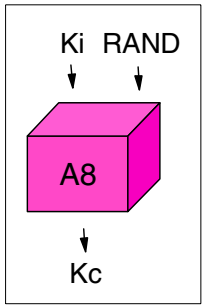


Figure 2-7 Generation of the encryption key in GSM

The traffic is then encrypted by using the generated key. It is important to note that the authentication key Ki is never transmitted over the air. Again, it is the AuC that actually computes the key Kc. The traffic from that moment on will be encrypted by using the generated key Kc and the GSM ciphering algorithm, called *A5*. The traffic is encrypted only over the radio interface, which means that the two parties that actually cipher and decipher the data are the mobile device

and the BTS. Then, the encryption algorithm A5 is not run by the AuC but rather by the BTS. For this reason, after the encryption key has been generated, the network forwards the Kc to the BTS of the area where the terminal is located, so that encryption and decryption can take place.

2.1.3 GPRS

General Packet Radio Service (GPRS) is the packet-based extension of the GSM network. GPRS is a fundamental step in the migration from GSM to 3G networks. This evolution path is represented in Figure 2-8.

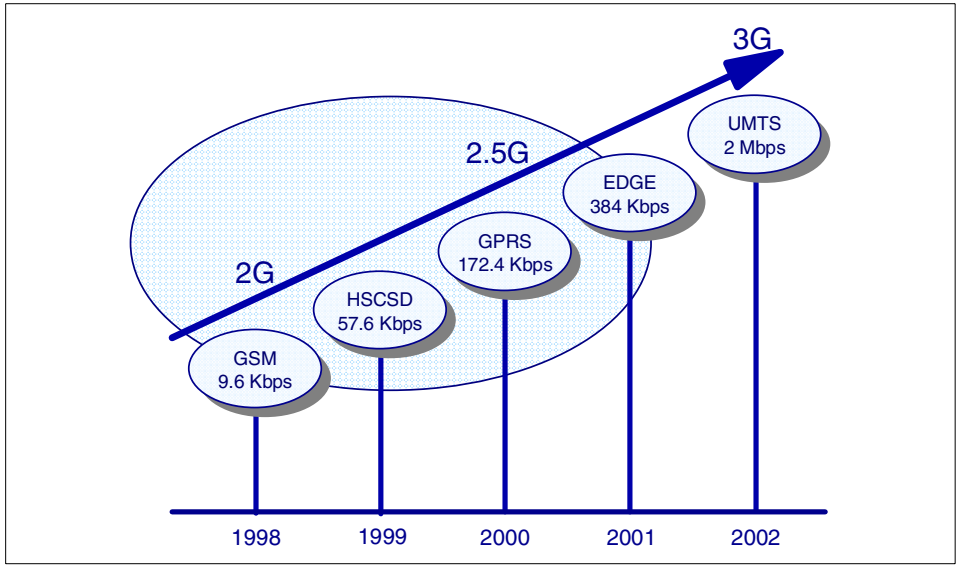


Figure 2-8 Evolution path from GSM to UMTS

GPRS can support data traffic over packet-based connections with a higher bit rate than GSM (up to 172.4 kbps). GPRS introduces three additional coding schemes (CS) for the data that is transmitted across the radio interface, with respect to the single coding scheme existing in GSM. These coding schemes, as seen in Table 2-3, provide different degrees of error correction and, consequently, different bandwidths (the values are computed using the whole MAC header). CS-1 is also used for the signalling channels in GSM and provides the highest error correction but the lowest throughput (9.2 kbps). On the contrary, CS-4 does not provide error correction at all, but ensures the highest data rate (21.55 kbps). In GPRS, timeslots can be bundled and allocated asymmetrically. While in GSM the user is allocated one timeslot in downlink and one timeslot in uplink, in GPRS more than one timeslot can be allocated per direction (one

through eight).The maximum throughput is computed by assuming CS-4 and the allocation of all the eight available timeslots. Furthermore, the number of timeslots in one direction can be different from the number of timeslots in the opposite direction.

Table 2-3 Transmission rates for GPRS

	1 Timeslot	2 Timeslots	8 Timeslots
CS-1	9.2 kbps	18.4 kbps	73.6 kbps
CS-2	13.55 kbps	27.1 kbps	108.4 kbps
CS-3	15.75 kbps	31.5 kbps	126 kbps
CS-4	21.55 kbps	43.1 kbps	172.4 kbp

Most of the emerging GPRS networks are providing CS-1 and CS-2, with one timeslot in uplink and two timeslots in downlink. One important consideration is that most GSM operators are building GPRS on top of the existing GSM infrastructure and are continuing to use GSM for voice traffic. This means that the radio resources must be shared between GSM and GPRS traffic and all timeslots at a time cannot be allocated. One important feature of GPRS and, more generally, of all packet-based networks is that the user is always connected to the network. Value-added services based on information push, such as online e-mail and remote monitoring, provide great advantages from this kind of network.

The GSM technology, even if based on a circuit-switched core network, is packet-based as far as the radio interface is concerned, because information is transmitted in bursts over the air interface. GPRS can be built on top of the existing GSM infrastructures, thus allowing a partial reuse of the already existing equipment. More precisely, the migration from GSM to GPRS can take place by leaving the GSM base station subsystem (for example, BTS and BSC) almost unchanged and by adding a packet-based switching subsystem to the GSM core switching subsystem (for example, MSC + GMSC). The GPRS network architecture is shown in Figure 2-9.

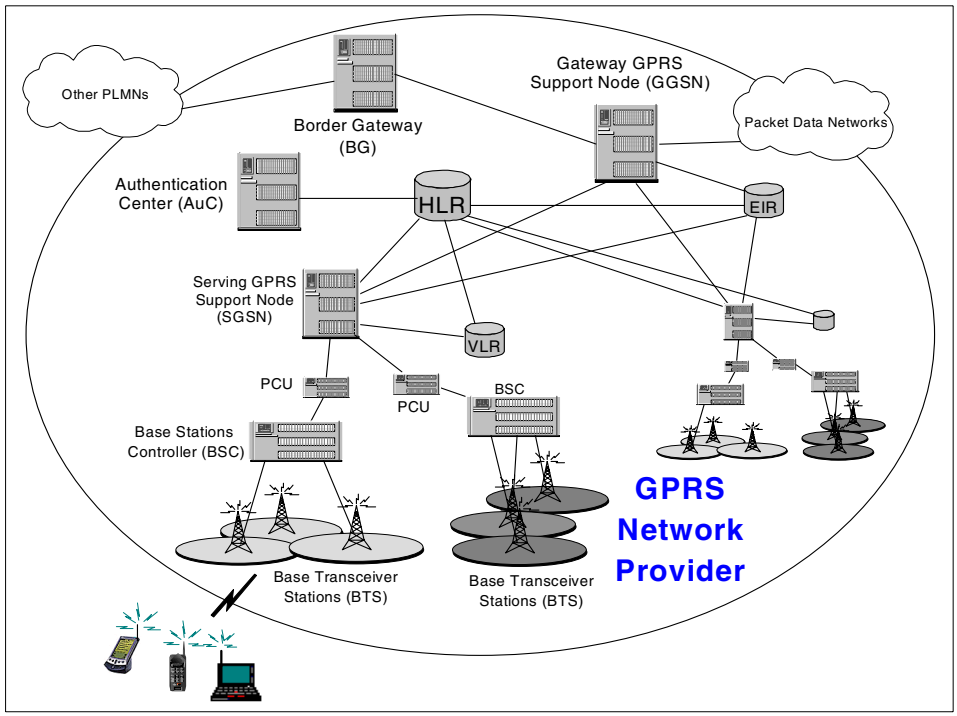


Figure 2-9 GPRS network architecture

The main differences between the GPRS network architecture and that of GSM can be seen in Figure 2-5 on page 28.

- **Packet Control Unit (PCU)**

  This new component performs an adaptation of content structure between the data packets received by the SGSN and the frame structure used by the BSC.

- **Serving GPRS Support Nodes (SGSN)**

  The SGSN is the packet-based equivalent of the MSC in GSM. It is able to route incoming data packets to the proper destination. Unlike GSM, where the communication is ciphered between the mobile station and the BTS, in GPRS the data is encrypted between the mobile station and the SGSN. This node must also provide ciphering capabilities. SGSN is also in charge of managing user mobility and has to collect accounting information. Finally, it must be capable of dealing with handover. If a user moves to another cell, all the undeliverable packets must be routed to the new cell. If the new cell belongs to a different SGSN, these packets must be routed to the new serving node.

- **Gateway GPRS Support Nodes (GGSN)**

  The Gateway GPRS Support Node (GGSN) interfaces the GPRS network with external packet data networks. It is also in charge of collecting accounting information and assigning IP addresses to GPRS mobile stations.

- **Border Gateway (BG)**

  The Border Gateway is the interface between a specific GPRS network and other mobile networks (PLMNs). It essentially provides security to protect the network.

Note: Even if GPRS has been defined as an extension of GSM, it will be used to provide packet-based support also to the IS-136 TDMA standard, which is popular in North and South America.

GPRS defines three different classes of mobile devices, as seen in Table 2-4.

Table 2-4 GPRS classes for mobile devices

GPRS class	Description
Class A	The mobile station can handle simultaneous circuit-switched and packet-switched connections (for example, it can transmit or receive GPRS packets with a GSM voice call in place).
Class B	The mobile station can support both circuit-switched and packet-switched connections, but cannot handle them simultaneously.
Class C	The mobile station supports circuit-switched or packet-switched connections.

In GPRS, every subscription has a well-defined *QoS* profile (Quality of Service). This will be negotiated in a handshake phase before any data transmission takes place. The QoS is defined as a set of four different parameters, as seen in Table 2-5.

Table 2-5 QoS profile parameters defined

QoS parameter	Description
Priority	It states if data transmission for this subscriber should have precedence over other transmissions.
Delay	It states how long it takes for a packet to cross the GPRS network.

QoS parameter	Description
Throughput	It indicates the mean and peak data rate available.
Reliability	It indicates the level of error correction provided.

As far as security is concerned, GPRS uses the same algorithms as GSM to generate the ciphering key Kc (for example, A8) and to generate the SRES value used for authentication (for example, A3). However, it uses a different encryption algorithm. As we have already stated, in GPRS encryption takes place between the mobile station and the SGSN.

2.1.4 Mobitex

The Mobitex technology was originally developed in Sweden in 1984 by Swedish Telecom Radio (now called Telia Mobitel) and tested by Erictel, which is a company owned by Swedish Telecom Radio and Ericsson Radio Systems. This network now operates in 23 countries. Mobitex is the network used in the U.S. by Palm.Net, as well as by many other wireless service providers. The architecture is depicted in Figure 2-10.

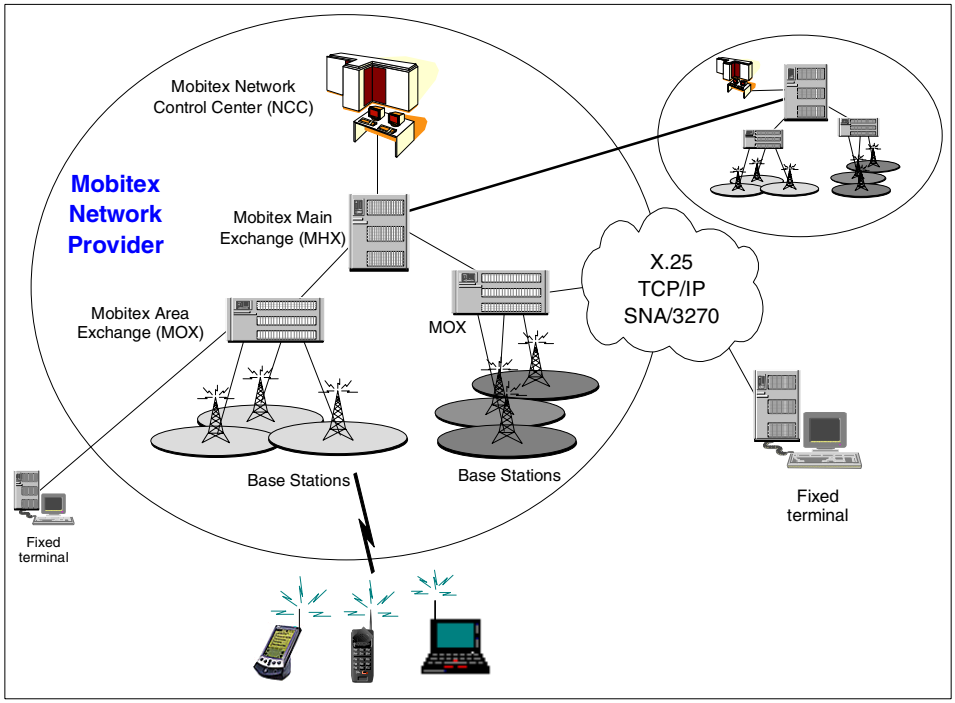


Figure 2-10 Mobitex network architecture

Mobitex in the US transmits from the base station on channels in the range of frequencies 935-940 MHz, and receives in the range of 896-901 MHz. The transmit and receive frequencies for any channel are always separated by 39 MHz. Mobitex in Europe has different frequency allocations for each country but they are all in the range of 410-459 MHz. Apart from this difference in radio carrier frequency, the European and North American systems are the same.

The data rate of a Mobitex wireless channel is 8 kbps. The network latency is relatively high and varies significantly. Mobitex provides a maximum message size of 512 octets. Both mobile devices and fixed terminals are treated equally in terms of addressing. Any entity can communicate with every end system in the Mobitex network. It is even possible to address end systems at other network providers if they are interconnected. Mobitex takes care of all the needed routing. So, different Mobitex networks may be interconnected and packets may be transferred between them, in the same way as you would establish international phone calls or communication between two telephone carriers. Mobitex mobile and fixed terminals are both identified and addressed by a Mobitex Access Number (MAN), which has an 8-digit value. MANs are unique worldwide and are assigned by the device manufacturer. Every Mobitex network provider has its own address range to assign MANs to fixed terminals. A Mobitex mobile device even keeps its MAN when roaming between different network providers.

Mobitex has been designed to be a messaging system, where small amounts of data have to be transmitted at irregular intervals. The transmission characteristics are not well suited for routing IP traffic. One problem is the high variance in network latency, which makes it hard for the network to decide whether a packet should be regarded as lost or just delayed (a bit longer).

Access to the Mobitex network from external host systems is provided by a number of different gateways, called *MOX*. The availability and definition of access methods to these gateways will vary from network to network, but normally includes:

- X.25
- TCP/IP
- SNA/3270

In a Mobitex network, the end systems exchange Mobitex packets, called *MPAKs*. Some MPAKs are exchanged between the terminals and the Mobitex system only. Others flow between the end systems and are routed by the MOX and MHX if they have to leave a MOX domain.

2.1.5 CDPD

Cellular Digital Packet Data (CDPD) is a wireless, packet-switched network technology. It uses the same frequencies as existing AMPS cellular services (for example, 800 MHz) and also shares the same core infrastructure. It was built on top of the AMPS infrastructure by adding the required capabilities for packet management and routing. Roughly speaking, CDPD is the packet-based extension of AMPS, which is circuit-switched, exactly as GPRS is the packet-based extension of GSM.

As for most packet-switched networks, charges are based on the amount of data sent rather than on the duration of the network connection. CDPD inherently uses Internet Protocol (IP) as the protocol for sending and receiving data. IP includes protocols that take care of such essential functions as authentication and encryption, and provides a maximum raw data throughput of 19.2 kbps. The network architecture is depicted in Figure 2-11.

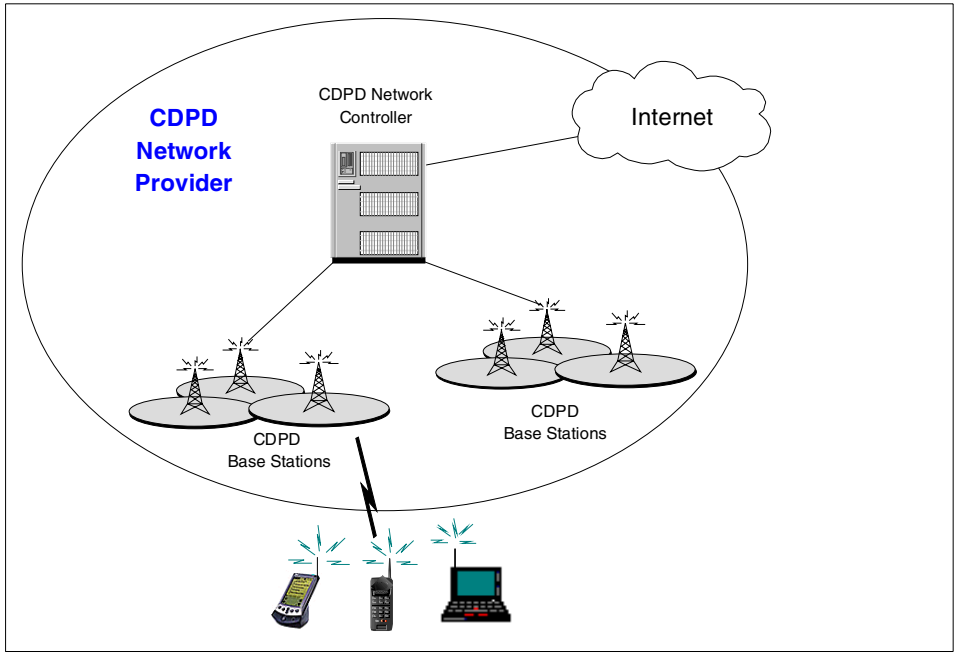


Figure 2-11 CDPD network architecture

The wireless CDPD Network Controllers, connected to the base stations, route the packet traffic between the CDPD phones and the Internet, and also manage the handover between cells. One of the most useful features of CDPD is the ability to find open voice channels and use them for data. CDPD base stations

monitor all the channels in the cellular network. When a base station receives a data transmission call, it picks out one of the unused channels and sends data over it. If a subsequent voice call needs that channel, the base station releases the channel and hops to another unused channel to continue the transmission.

Since CDPD relies on an AMPS cellular network, it is not available outside of North America and some Latin American countries. AT&T and Verizon in the U.S. offer wireless services over CDPD networks. However, roaming agreements between the two parties allow a huge coverage area. In this context, only AT&T offers WAP services over CDPD. Instead, Verizon provides WAP services over CDMA, which supports both voice and data traffic.

2.1.6 PDC

Personal Digital Cellular (PDC) is a Japanese cell phone standard that uses 800 MHz and 1.5 GHz frequency range. The data transmission rate is 9.6 kbps. Almost all the cell phones used in Japan are based on PDC.

2.1.7 IMT-2000

International Mobile Telecommunications-2000 (IMT-2000) is a 3G wireless system. IMT-2000 offers support for a wide range of mobile devices, includes links to terrestrial and/or satellite-based networks, and the terminals may be designed for mobile or fixed use.

Key features of IMT-2000 are:

- Support of different access networks, both terrestrial and satellite
- Multi-mode and multi-band terminals
- Virtual Home Environment (VHE)
- Very high data throughput
- Worldwide roaming capability
- Capability for multimedia applications

Note: VHE means that the user should have access as much as possible to the same set of services and the same look and feel regardless of the access network, the terminal capabilities, and the current location.

UMTS

The Universal Mobile Telecommunications System (UMTS) is the European implementation of the 3G wireless phone system. UMTS, which is part of IMT-2000, provides service in the 2 GHz band and offers global roaming and personalized features. UMTS was designed as an evolutionary system for GSM network operators, and offers impressive data rates of up to 2 Mbps. UMTS uses the W-CDMA technology. GPRS and EDGE are interim steps that will speed up wireless data for GSM.

For more information refer to: `http://www.umts-forum.org/`.

W-CDMA (DS-CDMA)

The Direct Spread - Code Division Multiple Access (DS-CDMA) specification is supported by Ericsson (Sweden) and Nokia (Finland). This technology will be used mainly in Europe and Japan. NTT DoCoMo will start the first IMT-2000 service in the world in mid-2001 using the W-CDMA specification. The data translation rate is 64 kbps for upstream and 384 kbps for downstream.

cdma2000 (MC-CDMA)

The Multi Carrier - Code Division Multiple Access (MS-CDMA) specification is supported by Qualcomm (US) and Lucent Technologies (US) and will be the North American standard. Compared to W-CDMA, it is easier for the carriers of cdmaOne to migrate their facilities or learn management know-how. The maximum data translation rate will be 14.4 kbps while fast moving, 384 kbps while slow moving, and 2 Mbps while at a standstill. The fast data transfer system called HDR (High Data Rate) developed by Qualcomm adopts a concept of best effort. One frequency is split into multiple transmissions. If the line is crowded, the speed will be less.

cdmaOne

Code Division Multiple Access (CDMA) is a specification of wireless communication. Voices from multiple users are transformed by multiplying different codes and transferred all together as one frequency. The receiver can detect only the sender's voice and decode it. The cdmaOne is one standard of 3G cell phones that uses CDMA protocol. It is used in North America and Asia. The data transmission rate is 14.4 kbps.

2.1.8 Wireless LANS

A wireless LAN is a local area network that transmits over the air, typically using an unlicensed frequency such as the 2.4 GHz band. A wireless LAN does not require lining up devices for line-of-sight transmission such as IrDA. Wireless access points (base stations) are connected to an Ethernet hub or server and

transmit a radio frequency over an area of several hundred to a thousand feet. This frequency can penetrate walls and other non-metal barriers. Roaming users can be handed off from one access point to another like a cellular phone system. Laptops use wireless modems that plug into an existing Ethernet port or that are self-contained on PC cards, while stand-alone desktops and servers use plug-in cards (ISA, PCI, etc.).

There have been numerous proprietary products on the market for home and office, but now most manufacturers are adopting the standard IEEE 802.11b, which defines a maximum data rate of 11 Mbps. Bluetooth and HomeRF are other home and small-office technologies that are expected to proliferate in the 2000-2002 timeframe. Such systems have a more limited range and do not support roaming. Small wireless LANs are sometimes called personal area networks (PANs), since one of their primary uses is to serve an individual connecting a laptop or PDA to a desktop machine.

2.2 Wireless protocols

Wireless protocols are used to connect mobile devices to the Internet. Many of the wireless protocols have defined architectures that optimize the use of the radio resource and also minimize the capabilities required for the device.

This section is organized into the following topics:

- HTTP protocol
- WAP
- i-mode
- Web Clipping
- Short Message Service (SMS)

2.2.1 HTTP protocol

The most obvious way of accessing Web-based applications from a mobile device is to use the same protocol and content type used when accessing the Internet from PC browsers over a wireline network connection. In this case, we use the HTTP protocol to retrieve standard HTML content, exactly as if we were accessing the Internet through a fixed workstation.

This approach, which is very suitable for wired connections, presents many limitations when adopted over wireless networks. In fact, when using TCP/IP based protocols (such as HTTP) over wireless mobile networks, you have to deal with the fact that radio networks usually have much less bandwidth and a higher

latency than local or wide area networks. Moreover, wireless connections are less stable in nature than wired connections and also unpredictable in terms of availability. TCP/IP-based protocols work well over wireless connections. However, the performance is slow.

Additionally, most mobile devices have limitations in display capabilities and may not be able to deal with all the features of full HTML. Sending full HTML to mobile devices can be useless in some cases, considering that some of this information will be ignored by the client browser. Due to the limited bandwidth and capability of mobile devices, simplified markup languages have evolved that are optimized for the limited bandwidth available and the limited capabilities of the mobile device.

Many alternative protocols and markup languages have been created for mobile environments. For example, the WAP technology (see 2.2.2, “WAP” on page 42) uses a protocol stack different from HTTP and a new markup language, such as WML. Moreover, it introduces a new scripting language for mobile devices, which is the equivalent of JavaScript in the Internet world. The i-mode approach (see 2.2.3, “i-mode” on page 46) is slightly different. It defines a new markup language, which is called Compact HTML (or more simply cHTML) and, as the name suggests, is a subset of HTML with a few special tags. However, unlike WAP, it still relies on the HTTP protocol.

The Palm Web Clipping solution (see 2.2.4, “Web Clipping” on page 52) is similar to i-mode in some respects. For example, they both use the HTTP/S protocols, but the page content is encoded according to a Palm proprietary format, called the Web Clipping Application (WCA) format. The content is retrieved from the content server in standard HTML and is then translated into WCA format by a proxy for proper rendering on the Palm device.

As the next-generation networks (for example, IMT-2000/UMTS) become available, which provide very high data throughputs, the HTTP/S protocols and HTML markup language become a much more viable solution for mobile devices.

2.2.2 WAP

The Wireless Application Protocol (WAP) is a standard for providing cellular phones, pagers, and other handheld devices with secure access to e-mail and text-based Web pages. Introduced in 1997 by Phone.com, Ericsson, Motorola and Nokia, WAP provides a complete environment for wireless applications that includes a wireless counterpart of TCP/IP and a framework for telephony integration, such as call control and phone book access.

WAP features the Wireless Markup Language (WML), which was derived from Phone.com's HDML and is a streamlined version of HTML for small-screen displays. It also uses WMLScript, a compact JavaScript-like language that runs in limited memory. WAP also supports handheld input methods such as a keypad and voice recognition. Independent of the air interface, WAP runs over all the major wireless networks in place now and in the future. It is also device independent, requiring only a minimum functionality in the unit so that it can be used with a myriad of phones and handheld devices.

WAP was developed because of the strong limitations of both mobile devices and wireless networks. Most wireless devices are often very limited in terms of display, processing power and available memory. Moreover, wireless networks themselves are characterized by limited bandwidth and high latency. In order for wireless devices to be able to access Internet content in a way similar to wireline PC browser clients, WAP was developed. The problems related to the use of TCP/IP over a wireless connection are described in 2.2.1, “HTTP protocol” on page 41. The WAP specification defines an architecture that optimizes the use of the radio resource and also minimizes the capabilities required for the device.

The main elements of the WAP specification are as follows:

- A Wireless Application Environment (WAE), which includes a microbrowser and is very well-suited for devices with poor capabilities.
- A new markup language called the Wireless Markup Language (WML), and a new scripting language called WMLScript.
- A new protocol stack, which is independent of the underlying network and is suitable for connections that have low bandwidth and are often unreliable.
- A secure protocol called Wireless Transport Layer Security (WTLS), which provides authentication and confidentiality.

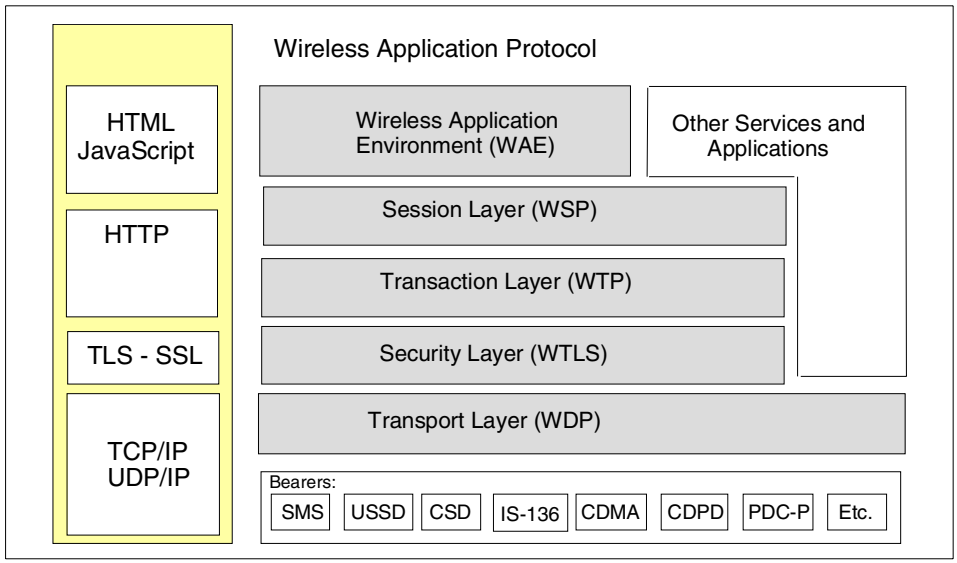


Figure 2-12 WAP stack vs Internet stack

Figure 2-12 shows the WAP stack (on the right-hand side) and how it compares to the Internet stack. The first thing to observe is that the WAP protocol does not make any assumptions about the underlying bearer service (for example, the network data service used to transport information). We can have WAP over a GSM circuit-switched data connection (CSD), over CDPD, and even over SMS. The transport layer, which is the lowest layer in the stack, is very similar to UDP. The topmost layer is the WAE, which includes WML and WMLScript that correspond respectively to HTML and JavaScript.

The WML markup language is based on XML. It can be viewed as a simplified version of HTML. It allows the user to enter data, select from a list, and display text and certain types of images. Example 2-1 shows a simple WML code sample.

Example 2-1 A simple WML code sample

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
   <card id="card1" title="A first example">
      <p align="center">
         Hello WAP World.
      </p>
   </card>
</wml>
```

WAP architecture

The communication between the client and the WAP gateway is based on the WAP protocol stack, whereas the link between the gateway and the content server uses standard HTTP/S. The WAP architecture is depicted in Figure 2-13.

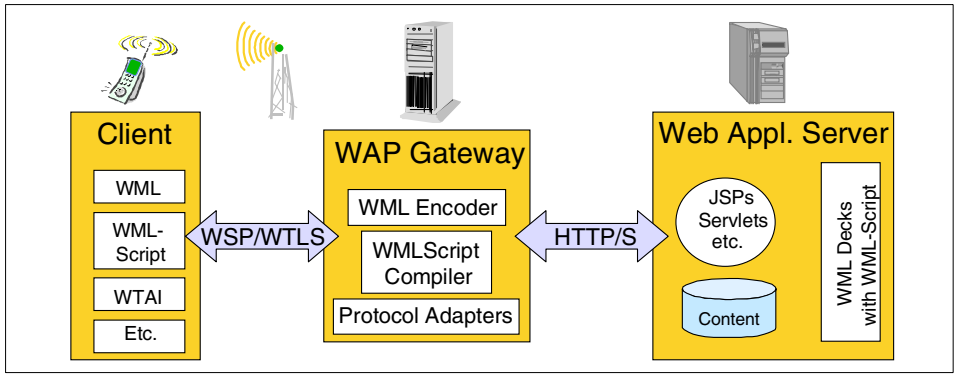


Figure 2-13 WAP architecture

The overall flow for a request coming from a WAP client is as follows:

1. The client sends a request for a WML deck by using the WAP protocol stack (for example, WSP, WTLS, etc.).
2. The WAP Gateway translates this request from the WAP protocol stack into the WWW protocol stack (for example, HTTP/S). The obtained request is then sent to the Web Application Server.
3. The Web Application Server generates the content to be returned to the client. The server can use advanced Web technologies, such as servlets or JSPs, to dynamically generate the response deck.
4. The content is returned as an HTTP/S response to the WAP gateway.
5. The WAP gateway encodes the content in a more compact format to optimize the use of the radio resource and then translates the response from the WWW protocol stack to the WAP protocol stack. Finally, the response is sent to the client.

If the WAP client sends a request for a WMLScript piece of code, this code is returned in a compiled format (to the client), in order to reduce the processing power required on the client device.

2.2.3 i-mode

i-mode is a wireless service developed by NTT DoCoMo in Japan. It is designed to provide mobile phone voice service, Internet and e-mail access. The i-mode protocol uses compact HTML (cHTML) as its markup language for the reasons that WAP use WML.

In order to develop i-mode applications, you will need to enter into a confidentiality agreement with NTT DoCoMo. The agreement will allow you access to the i-mode proprietary APIs to develop secure i-mode applications. NTT DoCoMo's i-mode service is predominately used in Japan. NTT DoCoMo has recently signed agreements with Telecom Italia Mobile in Italy, and AT&T in the US to provide i-mode service in 2002.

Carrier Internet access architecture

There are three major carriers for wireless connectivity of mobile phones in Japan:

- NTT DoCoMo
- KDDI
- J-PHONE

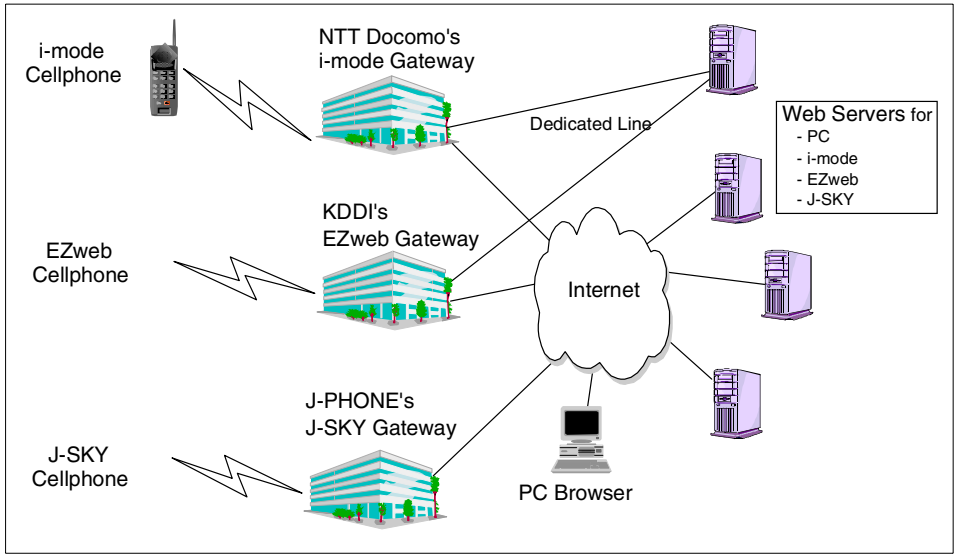


Figure 2-14 Mobile phones interaction with Web servers

Each of the carriers seen in Figure 2-14 has been providing Internet connectivity service for their mobile phone users since 1999. NTT DoCoMo provides i-mode service to approximately 35,000,000 customers, of which 20,000,000 customers have Internet access.

Each carrier has a gateway center to translate the wireless transmission from the mobile phone to TCP/IP protocol so that it can be sent over the Internet. The i-mode service provided by NTT DoCoMo has a basic fee of 300 Yen (approximately 2.5 USD), and bills customers and additional charge based on the amount of packets that the user sends and receives from the gateway. For example, 128-byte packet transferred costs 0.3 Yen (0.0025 USD).

The carrier provides Internet e-mail service with an address such as `xxxx@docomo.ne.jp`. There is a one-to-one correlation of e-mail address and phone number. The outbound e-mail is transferred through the gateway to the Internet. The inbound e-mail is routed through the gateway and notification is sent to the mobile phone that an e-mail has been received.

The concept of a SIM card provided by WAP is not used by i-mode or other protocols or services in Japan. If you want to change from your old phone to a new mobile phone and you are using the same carrier, you will have to choose from one of the following options:

- Use the former phone number and e-mail address.
- Use a new phone number and e-mail address.

If you want to change the mobile phone to a different carrier service, you have to change your phone number as well as your e-mail address.

Note: The concept of a SIM card will be provided with the rollout of the new IMT2000 mobile phone provided by NTT DoCoMo.

Official Site and Voluntary Site in Japan

Each carrier in Japan has an Official Site service. The content provider (Web site or commerce site) needs to register with the carrier to become an Official Site. Once the content provider is approved as an Official Site, they gain access to many benefits from the carrier. For example, Official Site content providers are listed on the home page of mobile phones serviced by the carrier. The carrier acts as a portal site when the user of the mobile phone accesses the Internet. This feature is always turned on and is not customizable by the user.

It is not easy to type long URLs on a mobile phone. The user can connect to the Official Site easily by navigating the menu, thus giving the content provider a much higher chance of the site being accessed.

Another benefit of the Official Site is that the carrier, rather than the content provider, collects access fees to pay for content. The carrier sends a bill to the users which includes the usual calling fee, Internet service fee (including e-mail service), and an access fee to pay for the content served based on the packets transferred. Having payment services provided by the carrier makes it easy for content providers to engage in m-commerce without having to address issues related to interfacing with many payment systems and technologies. Users are billed a fee for accessing content provider for m-commerce. For example, NTT DoCoMo includes a 9% surcharge on all fees collected from the content providers.

The third benefit of an Official Site is security. For example, i-mode provides a dedicated line connection service between the service provider i-mode gateway and the Web server of the Official Site. Content providers such as banks or securities firms for trading stock commonly use this service to get a highly secure connection.

The carrier can use the mobile phone unique identifier (in some cases supplied by the carrier gateway) and the IP address of the carrier gateway to build a secure system. WCS V5.1 uses this information for session control between the mobile phone and the WCS server. For more information on WCS V5.1 m-commerce session control, refer to 4.3, “Session control” on page 92.

Table 2-6 provides a summary of the differences between Official and Voluntary Sites for i-mode.

Table 2-6 The difference between Official Site and Voluntary Site for i-mode

Title	Official Site	Voluntary Site
Number of sites	1600	39000
Navigate by	Menu list	URL/other portal site
Content	Acceptable for public policy	Anything
Link to other site	Restricted	As they wish
Payment for contents	Provided	Not provided
Dedicated line to Web server	Provided	Not available
Mobile phone device information	Provided	Not available
IP address of gateway	Provided	Not available

NTT DoCoMo does not charge extra registration fees for the Official versus Voluntary Sites. The business model of NTT DoCoMo is based on usage. Specifically, users are charged a fee according to how many packets they have transferred through the gateway. The more popular a Web site is, the more profitable the site will be for NTT DoCoMo.

If content providers want to become an Official Site, they are required to submit a business case proposal, which explains the business potential, uniqueness, and the advantage to the other Official Sites. The carrier will examine the proposal and decide if the content provider is worthy of an Official Site or not.

Voluntary Site access is free. All that is needed to make content available for a Voluntary Site is the appropriate markup language. One disadvantage is that you cannot get any mobile device-related information from the carrier. It is more difficult for WCS V5.1 to provide session control without this information. In Japan, if you want to build an m-commerce Web site, we strongly recommend that you try to become an Official Site.

Note: A new model of mobile phone called the 503i Series from NTT DoCoMo in Japan has the capability to transfer its own device identifier to the Web server. This feature allows Voluntary Sites to be accessed with a unique identifier critical for security, without needing the carrier gateway to provide this service.

NTT DoCoMo 503i Series: Cell phone with Java

The 503i Series supports two major new features: Java and SSL. The Java platform has been developed to support the 503i Series. In addition to the i-mode extended libraries, it contains the following:

- Java2 platform Micro Edition (J2ME)
- Connected Limited Device Configuration (CLDC)

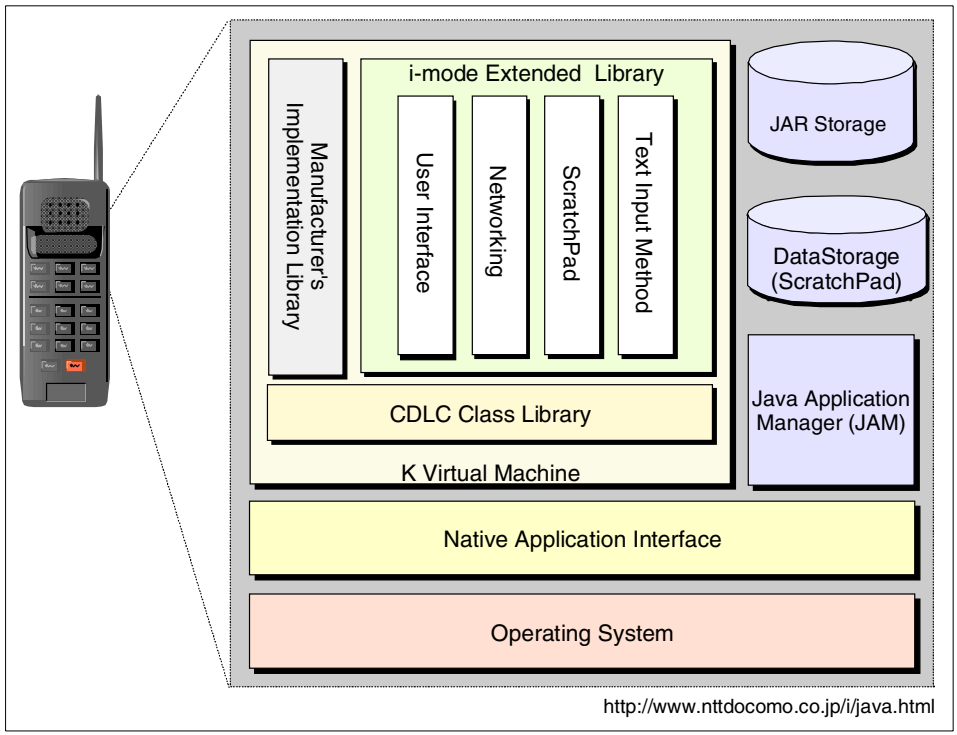


Figure 2-15 Java architecture for NTT DoCoMo 503i Series

The Java architecture of the NTT DoCoMo 503i Series mobile phone is depicted in Figure 2-15. Users of the 503i Series can download programs from Web sites via the HTTP protocol. The downloaded program is called iAppli, and is executed as a Java applet.

The device is small, so there are some limitations and considerations for using Java applets on 503i Series mobile phones.

- Java applet

  Once the program is downloaded, it is not stored in the browser cache but stored permanently into a type of storage called a ScratchPad. It is possible to launch the applet from the ScratchPad when the user is not connected to the Internet via i-mode. NTT DoCoMo advertises that the user can store at least three programs on the 503i mobile phone. Each program is required to be less than 10 KB as a JAR file. The data size must be less than 5 KB in the ScratchPad.

- Semi-multimedia

  Sound files called i-melody and animation GIF files are supported.

- Scheduler

  It is possible to launch the program at a specified date, at a certain interval, and access a Web site using a user agent. The scheduler is used to enable agent applications, such as a stock price chart or a weather forecast.

- Security

  To protect the user from a malicious program, Java applications cannot access the local address book or dialer function of the mobile phone. We have listed the restrictions to Java programs in regards to security, which is controlled by Java Application Manager or JAM:

  – A Java program cannot launch or override another Java program.

  – A Java program cannot access the other ScratchPads.

  – A Java program cannot connect to the Web server, except the one from which the program was downloaded.

  – A Java program can be terminated by the user.

- Compatibility consideration

  There is no compatibility with the Mobile Information Device Profile (MIDP) for the J2ME platform.

Note: In order to develop i-mode applications, you must enter into a confidentially agreement with NTT DoCoMo. This agreement will allow you access to technical information required to develop secure i-mode solutions.

- Refer to the following URLs for information about i-mode provided by NTT DoCoMo:

  `http://www.nttdocomo.com`
  `http://www.nttdocomo.com/imode`

  Some information is available on the NTT DoCoMo Web site; however, this information is only available in Japanese.

- For questions regarding implementing m-commerce using WCS V5.1 for i-mode mobile phones, contact IBM Japan at:

  `http://www.ibm.com/jp/contact`

For more detailed information or considerations refer to the *i-mode Java Contents Development Guide* found at the following URL:

`http://www.nttdocomo.co.jp/i/java.html`

Note: The *i-mode Java Contents Development Guide* is only available in Japanese.

2.2.4 Web Clipping

Web Clipping is a proprietary technology developed by Palm. The main elements that constitute the Web Clipping architecture are the Palm device, a Web Clipping proxy server, and the content server. In order to support Web Clipping, a special piece of software called a Clipper must be present on the Palm device. This application is a built-in component of all Palm VII devices. However, some third-party software (for example, OmniSky) can be used to make Web Clipping applications available also on Palm V and IBM WorkPad devices. The Clipper is able to interpret and properly render a proprietary format called Web Clipping Application (WCA) format. This format makes it possible to reduce the amount of data transferred over the air and the required storage on the Palm. The content that is retrieved from the content server is in HTML format. The Web Clipping proxy is in charge of translating the HTML content into this proprietary WCA format. There are many available proxies that can be used for this, all hosted by Palm.Net. Web Clipping uses the HTTP/S protocols.

There are two main components to consider when writing Web Clipping applications:

- The Web Clipping application has to be installed on the device.
- The server side application, which returns the result pages to the device.

A Web Clipping application is like a small Web site stored locally. The starting (or index) page usually provides a form, or list of links, which are the gateways to the live data provided by the server.

Result pages (clippings) are returned by the server side application as a response to the request from the Web Clipping application. The result pages are written in HTML.

Web Clipping proxy server

A key component of the Web Clipping solution is the Palm Computing Web Clipping proxy server that resides at the 3Com Corporation's data center. The Web Clipping proxy server is responsible for converting the standard Internet protocols and content from a Web page into a form that is tuned for transmission across a wireless network and for display on a small device.

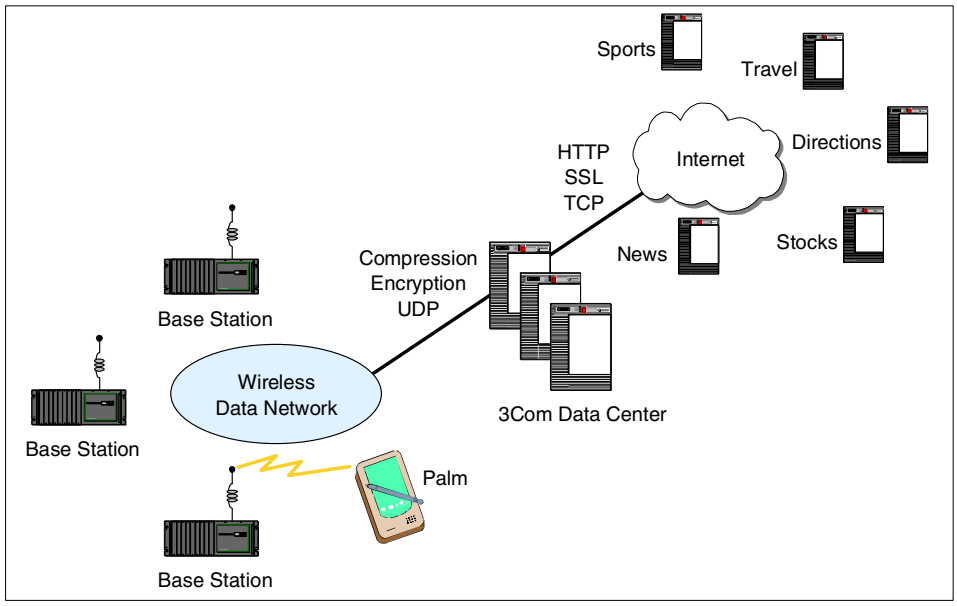


Figure 2-16 Web Clipping proxy server

As shown in Figure 2-16, the Web Clipping proxy server uses the standard Internet protocols (TCP, HTTP, SSL) to access Web servers. To ensure compatibility, the requested pages are in HTML format.

The Web Clipping proxy server implements a reliable layer over the UDP protocol to talk to the Palm device; this protocol reduces latency and conserves battery power relative to using TCP.

In Web Clipping, encryption and authentication between the handheld and the Web Clipping proxy server is performed by Elliptic Curve Cryptography from Certicom Corporation, which offers extremely high levels of security at small key sizes. On the server side, the high-strength SSL is used for encryption and authentication between the Web Clipping proxy server and Web servers providing HTML content.

Note: For more detailed information about Web Clipping application development for Palm devices, refer to the following URL:

`http://www.palm.com/dev`

2.2.5 Short Message Service (SMS)

Short Message Service (SMS) provides two-way alphanumeric paging messages that can be sent to and from mobile phones.

This section is organized as follows:

- Messaging
- Short Message Service (SMS)
- WCS V5.1 messaging
- SMS implementation

Messaging

An m-commerce solutions usually requires an online client connection to interact with the commerce application. There are other possibilities where an asynchronous connection or synchronization connection are possible. These cases are beyond the scope of this redbook.

During time periods when a user does not have access to a PC browser, and a user needs to be notified to alert the user of an event, a message can be sent to the user's mobile phone (for example, an auction when someone has a higher bid). The messaging subsystem of WCS V5.1 allows the generation of a message when a specified event occurs.

Short Message Service (SMS)

Short Message Service (SMS) messages are two-way alphanumeric paging messages that can be sent to and from mobile phones. SMS messages are transmitted over the mobile phone's air interface, using the signalling channels so there is no delay for call setup. SMS messages are stored by an entity called the Short Message Service Center (SMSC) and sent to the recipient when the subscriber connects to the network. The number of a cooperative SMSC must be known to the sender when sending the message. Short messages can be mobile terminated (MT) or mobile originated (MO). Mobile -terminated messages are the ones that arrive at the phones; mobile originating messages are sent by a mobile subscriber. Networks may support either, both, or neither one of these.

A service similar to GSM SMS can also be found in other mobile phone systems. To keep everything simple, this redbook specifies a unique scheme specifier for SMS messages in the GSM system; other systems may use other scheme specifiers.

The SMS messages cannot be longer than 160 characters, in case the message has to be formatted to fit into this short text.

WCS V5.1 messaging

The WebSphere Commerce Suite V5.1 messaging system is capable of managing all aspects of defining and sending messages generated within Commerce Suite. It allows you to control the manner in which administrators, customers, and the back-end systems are notified of various events, such as customer orders or system errors.

A configuration of the outbound messaging system is performed using the administration console. Through the administration console, you can determine what transport methods are to be used, configure them to your specifications, and assign message types to transport methods. The messaging system can transport messages via e-mail using SMTP and plain files, and you can also configure the system to send messages to the back-end system using MQSeries.

The messaging environment features include the following:

- Composition services, which give you the ability to customize the messages using predefined JSP templates.
- Support for multiple message transmissions, which allows you to send a single message through more than one transport.
- Multiple notifications can be sent over the same transport. This is useful for sending a broadcast e-mail to multiple recipients. The content of a broadcast e-mail can vary for different recipients.
- Support for the following three forms of processing:
  - Transacted processing for messages that should only be sent upon successful completion of the current transaction.
  - Immediate processing for messages that must be sent at the time the event takes place in Commerce Suite. With this type of communication, the message is sent whether the transaction is committed or not.
  - Request-reply processing for messages that require a response from the back-end system.

SMS implementation

There are a few different approaches to creating an SMS message with these tools:

- Using a specially formatted e-mail, and then sending it to an e-mail-SMS gateway.
- Using an HTTP-SMS gateway, where SMS is sent through an HTTP connection.

- Using an SMS server, which can connect via a modem to the SMS Center and send out messages received from the application server.

The first solution is much easier, and requires less knowledge of SMS messaging, while the other solutions require more programming.

Since most of the service providers allow the users to send SMS in e-mail, and WebSphere Commerce Suite V5.1 provides e-mail messaging, the e-mail-SMS gateway solution is the obvious choice.

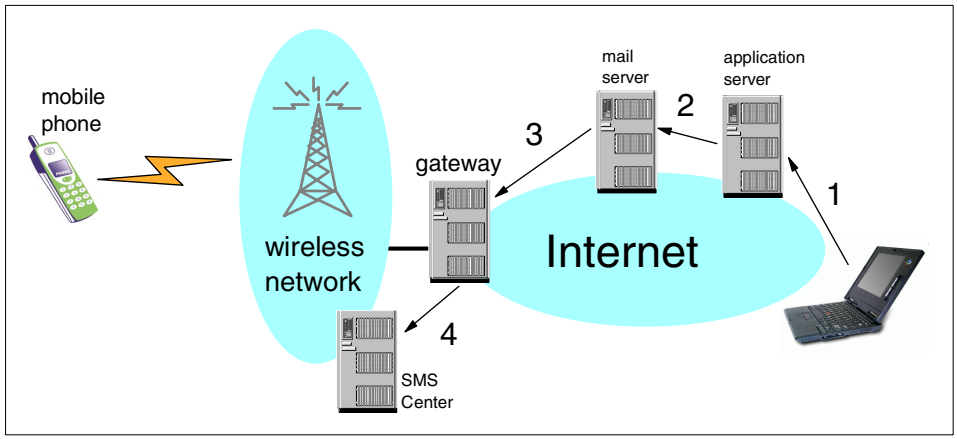


Figure 2-17 E-mail-SMS gateway scenario

Figure 2-17 represents a simple e-mail-SMS gateway scenario, with the following steps:

1. The client sends a request to the application server, which invokes a messaging action.
2. The event starts the messaging procedure, where the message is generated and sent through the network as an e-mail using SMTP.
3. The mail server delivers the e-mail to the recipient, which is the e-mail-SMS gateway provider.
4. The gateway sends the appropriate message to the SMS Center, which delivers the SMS to the mobile device.

Two major steps are required to set up the messaging service on the WebSphere Commerce Suite server:

1. Configure the WCS V5.1 messaging services

   For detailed instructions, refer to 7.4.1, “Configure the WCS V5.1 messaging services” on page 155.

2. Defining the message content

   For detailed instructions, refer to 7.4.2, “Define the message content” on page 156.

2.3 Mobile devices

We have placed the mobile devices into three categories to describe the specific features that they offer. In practice, there is an ever-growing number of hybrid devices. For example, the Ericsson R380 WAP phone also has the capabilities of a PDA.

This section is organized as follows:

- Mobile phones
- Wireless PDAs
- Wireless laptops
- Mobile device pros and cons

2.3.1 Mobile phones

When we refer to a mobile phone in this redbook, we are talking about mobile or wireless phones that have a microbrowser to access Internet content. Other names for a mobile phone include cell phone and wireless phone.

A standard mobile phone includes the following capabilities:

- Voice
- Messaging (SMS or WAP push)
- Data (access Internet content via a microbrowser)

Note: For the purposes of this redbook, we have excluded analog cell or mobile phones that do not provide any Internet browsing capability.

There are two major protocols that are used for mobile phones capable of Internet browsing: WAP and i-mode.

2.3.2 Wireless PDAs

A Personal Digital Assistant (PDA) is a handheld computer with a wireless interface that serves as an organizer for personal information. PDAs often have a pen-based stylus to tap selections on menus and to enter printed characters. The unit may also include a small on-screen keyboard that is tapped with the pen. Data is synchronized between the PDA and desktop computer via cable or wireless transmission.

We are interested in PDAs that have wireless transmission capability and include a Web browser. The major operating systems for PDAs are Palm OS, EPOC, and Windows CE. Table 2-7 displays the major wireless PDAs, operating systems, device manufacturers, and protocols used.

Table 2-7 PDA operating systems and device manufacturers

PDA type	PDA operating system	PDA Manufacturer	Protocol
Palm	PALM OS	Palm V, Palm VII	HTTP, WAP, Web Clipping
		IBM WorkPad PC	HTTP, WAP, Web Clipping
		Handspring Visor	HTTP, WAP, Web Clipping
PocketPC	Microsoft Windows CE	Compaq iPac	HTTP, WAP
		HP Jornada	HTTP, WAP
		Casio E-125	HTTP, WAP
EPOC	Symbian EPOC32	Ericsson R380	WAP
		Nokia 9210 Communicator	WAP
		Psion Series 5mx, Series 7, REVO, REVO PLUS	WAP

Palm

The base for this solution is not the device, but the operating system (OS). The OS called PALM OS is used with a wide range of devices, including the Palm VII, the IBM WorkPad, and HandSpring Visor. This device has built-in mobile capabilities, and works as a mobile phone with a data connection.

Like many other operating systems on the market, the Palm OS has different editions. The most widely used editions are: V3.1, V3.3 and the latest, V3.5. All the versions are improvements on the previous versions. It is important to notice that some of the software relies on a specific version of the OS.

Palm OS has the capability of running several different browsers to access Internet content, including:

- HTTP browser
- WAP browser
- Web Clipping browser

HTTP browser

The HTTP browsers on Palm OS are like any other HTTP browser for accessing Internet content, such as Netscape Navigator or Microsoft Internet Explorer.

From the m-commerce application point of view, there are some key features required by the browser:

- Secure Socket Layer (SSL)
- Forms
- Cookies

There are several HTTP Web browsers available for Palm OS as follows:

- Intellisync Browse-it

  This browser features impressive display capabilities, even for the small PDA screen. The browser supports SSL and cookies, and the proxy server is configurable. The browser requires a proxy server from the service provider. The software and the proxy service are available for free.

- Handspring Blazer

  This is a great browser for a small device such as a PDA. It has all the required capabilities, such as images (b/w, gray, color), security (requires proxy at the provider), cookies, and HTTP proxy. It supports HTML, WAP (WML/HDML), and cHTML. The browser and the proxy service are free from Handspring.

- AvantGo AvantGo

  This browser, just like the previous, supports all the necessary functions. The presentation is average. It does not support a proxy server, but this is not required. AvantGo is a well-known and long-running product that is available for free. It requires a proxy server for content providing. The service is free.

- OmniSky OmniSky

The OmniSky browser is basically the same as the Palm VII built-in Web Clipping viewer.

- Qualcomm EudoraWeb

  This is the only browser at this time that uses a true SSL connection. Unfortunately, the software is not free. It comes with all the required features, and supports a proxy. Using the software with the Palm OS Emulator, the Name Server IP address has to be defined (use the **`nslookup`** command at the command prompt to determine the Name Server IP).

There are other HTML Web browsers, but they are not useful in this case, since they do not support secure connections.

Note: In Chapter 16, “Palm implementation sample” on page 279, we provide design and development guidelines as well as sample code for implementing m-commerce applications for Palm HTML-based browsers.

WAP browser

Palm uses the same WAP browsers as WAP phones. The main difference is the size of the screen.

Some WAP browsers include the following:

- EdgeMatrix WAPman

  This is a commercial WAP browser with WTLS security features.

- 4thpass Kbrowser

  This is a browser that does not require WAP gateway.

These browsers provide a big screen with WAP wireless access. Considering the air-time and the small amount of content, it is reasonable to use WAP browsers on a Palm device.

Web Clipping browser

To achieve the goals of long battery life, low service cost and Internet-like performance with low bandwidth, Palm Computing took a different approach to accessing information on the Web. Browsing does not make sense for a handheld device with a small screen and low bandwidth. The Web Clipping solution is like clipping an article out of a newspaper to get the part that is needed, and nothing more.

Proxy servers

Some of the above-mentioned browsers require a server called a proxy, which is a server between the Web browser client and a Web server. The proxy is necessary for security purposes. These small devices are usually not capable of calculating heavy encryption keys, such as SSL. Therefore, the device uses a light (but strong enough) encryption method between the client and the proxy; then the proxy connects to the Web server using SSL.

The problem is that the client cannot do anything with server certificates, because the proxy handles the security. Usually proxies only accept certificates predefined by the service provider (which runs the proxy). If the destination Web server certificate is not in the proxy database, the client cannot connect to the site securely. This is a common problem during the development of Palm Web applications.

PocketPC

PocketPC devices use the Microsoft Windows CE (WinCE) operating system. The latest version (3G) is called PocketPC, which is similar to the well-known Windows operating system, but optimized and developed especially for PocketPC mobile devices.

HTTP browser

WinCE/PocketPC is capable of running more advanced Web browsers, such as the WinCE/PocketPC version of Internet Explorer called PocketIE. The software is freely available for download from the Internet.

The browser supports the required features to access the m-commerce sites with support such as SSL, cookies, and proxy server configuration ability.

If the recommended design method is using simplified HTML, cut down the content - avoid using big images, nested tables, JavaScript, and Java.

WAP browser

Since WinCE/PocketPC devices are powerful enough to run sophisticated Web browsers, the availability of WAP browsers is very limited. Eventually an application may require a WAP browser on a WinCE/PocketPC device, but it is not reasonable because there are better applications for wireless communication.

Some examples of WAP browsers for WinCE/PocketPC include the following:

- Ezos's EzWAP V1.0 for all Windows platforms
- Ezos's EzWAP V2.0 for PocketPC only

EPOC

Symbian's EPOC is an emerging operating system in the wireless PDA market. Some examples of manufacturer devices that use EPOC include the following:

- Psion Series 5mx, Series 7, REVO, REVO Plus
- Ericsson R380
- Nokia 9210 Communicator

You may notice that the devices are very different from each other. The OS for these devices is EPOC, but the applications are different.

HTML browser

The most powerful browser for EPOC is the latest version of the freely available browser called OPERA. This browser was originally available for PCs.

WAP browser

Since the Ericsson R380 and the Nokia 9210 are mobile phones or communicators, they have built-in WAP browsers.

The Pocket PC does not replace Windows CE. The Pocket PC is the hardware device, and Windows CE is the operating system. The Pocket PC uses a customized version of the Windows CE 3.0 operating system, built by Microsoft and used specifically in Personal Digital Assistants (PDAs), such as the Compaq iPaq and the Hewlett-Packard Jornada. While this customized version is only used in PDA-type devices, Windows CE 3.0 can be used in a wide variety of devices including industrial automation devices, Internet access devices, Web terminals, kiosks, consumer electronics, or retail and point-of-sale devices.

2.3.3 Wireless laptops

This category includes laptops, notebooks, or portable PC browser clients that have a wireless interface to the network for Internet access. These clients use standard TCP/IP protocols and a standard browser, such as Netscape Navigator or Microsoft Internet Explorer. The wireless connection is usually much slower than wireline-based network clients. An example can be an IBM ThinkPad T20 with a GSM mobile phone connected through the infrared port.

2.3.4 Mobile device pros and cons

Compared to a personal computer, the mobile device is small and portable. This portability is an advantage over the PC in usability, accessibility, and costs. However, there are disadvantages when using a mobile device instead of a PC. We have listed some of the pros and cons of using a mobile device.

Pros of mobile devices

- Portability

  The user can receive information anytime and anywhere by e-mail or a direct phone call. This enables people to receive information instantly. Combined with GPS (Global Positioning System) capability, you can always be reached.

- Easy operation

  Mobile devices can be switched on instantly. The user can operate a cell phone with a single hand, even with one finger or thumb. Also, a power cable is not needed.

- Low cost

  Mobile devices consume very little power, and are relatively cheap in comparison to a PC browser client.

Cons of mobile devices

- Quality of line connection

  The connection speed to the Internet is relatively slow. The connection can sometimes be disconnected when a user is moving outside because of interference, being out of range, or out of the frequency.

- Security

  If your mobile phone is lost or stolen while you are connected to a commerce site, the person with the phone can make purchases at your expense.

  Typically, the password is not visible when you enter it on a PC (* is shown). On a mobile phone, the four or six digits of numeric character are often used (lowered security).

- Poor user interface

  It is difficult to enter characters from a standard keypad on a mobile device. Some mobile phones requiring pushing the same key several times to represent different characters in the alphabet.

  The key layout is different from device to device and there is no mouse operation supported as in Windows.

- Small display and cache
- No attachment file support for e-mail (cell phone)
- No cookie support (certain devices)

  It is not easy to maintain sessions between a mobile device and Web servers without a cookie. Some alternative techniques should be used.

- No JavaScript support

Note: WAP does provide WMLScript, which offers capability similar to JavaScript.

- URL length limitation

  On some devices, the maximum length of a URL is 400 bytes.

2.4 Content and markup languages

Content is the information presented to the user on a Web page. This section provides information about the markup languages used to render content for Web pages.

Markup languages are a set of labels that are embedded within text to distinguish individual elements or groups of elements for display or identification purposes. The labels are typically known as tags. Markup languages identify elements within a continuous stream of text.

Note: The programming model in WebSphere Commerce Suite V5.1 uses JSPs for view of the model-view-control model. In this case the view is done using a JSP with a markup language such as HTML, XML or WML embedded within the JSP to create the content for the Web page.

Our intention is not to document how to use the following markup languages, but to provide insight on how they are used within the context of mobile computing and m-commerce:

- HTML
- HDML
- WML
- cHTML
- XML
- XHTML

HTML

HyperText Markup Language (HTML) is the most widely used markup language on the World Wide Web. Web pages are built with HTML tags, or codes, embedded in the text. HTML defines the page layout, fonts, and graphic elements, as well as the hypertext links to other documents on the Web. Each link contains the URL, or address, of a Web page residing on the same server or any server worldwide, hence the name World Wide Web.

HTML is not a programming language like Java or JavaScript. It could be considered a presentation language. HTML is derived from the Standard Generalized Markup Language (SGML), which is widely used to publish documents. HTML is an SGML document with a fixed set of tags that, although they change with each new revision, are not flexible.

HTML is the standard markup language used within the JSPs of WebSphere Commerce Suite V5.1 and WebSphere Application Server. Initially, HTML was not used in mobile computing because the network bandwidth and the capability of the device were so poor that HTML was considered too inefficient. Since the 2.5G and 3G wireless networks have been implemented, network bandwidth is no longer an issue. The new more powerful mobile devices are capable of using the HTTP protocol and HTML.

Note: We provide a sample m-commerce implementation using HTML for:

- m-commerce Direct in 16.1, “Palm HTML browser implementation” on page 280.
- m-commerce using WTP in 19.3.1, “HTML content using WTP” on page 307.

HDML

Handheld Device Markup Language (HDML) is a specialized version of HTML. Designed to enable wireless pagers, cell phones, mobile phones and other handheld devices to obtain information from Web pages. HDML was developed by Phone.com (formerly Unwired Planet) before the WAP specification was standardized. It is a subset of WAP with some features that were not included in WAP.

HDML is still widely used in the US and has several million users in Japan, although the dominant markup language and service are cHTML used with i-mode. The UP.Browser developed by OpenWave Phone.com provides support for HDML markup.

Note: We provide a sample m-commerce implementation using HDML in Chapter 14, “HDML implementation sample” on page 265.

WML

Wireless Markup Language (WML) is a tag-based language used in the Wireless Application Protocol (WAP). WML is an XML document type allowing standard XML and HTML tools to be used to develop WML applications. It evolved from Phone.com's HDML, but WML is not a superset of HDML. Certain HDML features are not found in WML.

Note: We provide a sample m-commerce implementation using WML in Chapter 15, “WAP implementation sample” on page 271.

cHTML

Compact HTML (cHTML) is a more efficient variation of HTML specifically designed for use by the i-mode wireless service. Information on cHTML can be found at the following URLs:

```
http://www.nttdocomo.com/i/index.html
http://www.nttdocomo.co.jp/i/tag.html
```

XML

Extensible Markup Language (XML) is an open standard for describing data from the W3C. It is used to define data elements on a Web page and business-to-business documents. It uses a tag structure similar to HTML; however, whereas HTML defines how elements are displayed, XML defines what those elements contain. HTML uses predefined tags, but XML allows tags to be defined by the page developer. Thus virtually any data items, such as product, sales representative and amount due, can be identified, allowing Web pages to function as database records. By providing a common method for identifying data, XML supports business-to-business transactions and is expected to become the dominant format for electronic data interchange (EDI).

Since its introduction, XML has been hyped tremendously as the panacea for e-commerce. The human-readable XML tags provide a simple data format, but the intelligent defining of these tags and common adherence to their usage will determine their value. For example, commercial XML (cXML) from Ariba and Common Business Library (CBL) from Commerce One are some of the first XML vocabularies for business data. DSML is a set of XML tags that defines the items in a directory. XML tags are defined in an XML schema, which defines content type as well as name. XML tags can also be described in the original SGML DTD format, since XML is a subset of the SGML language. There are several Web sites that provide repositories for publishing and reviewing XML schemas.

Unlike HTML, which uses a rather loose coding style and which is tolerant of coding errors, XML pages have to be well formed, which means they must comply with rigid rules.

Note: We provide a sample for using XML in 19.3.2, “XML content using WTP” on page 309.

XHTML

Extensible HTML (XHTML) is a combination of HTML 4.0 and XML 1.0 into a single format for the Web. XHTML is expected to become the standard format for Web pages. XHTML also makes it possible for Web pages to be developed with different sets of data, depending on the type of browser used to access the Web. Increasingly, handheld devices are used for the Web that must download abbreviated pages, because they do not have screen displays large enough and capable of handling the graphics.

2.5 Wireless service providers

A wireless service provider is an organization that provides wireless services to its customers, including cellular services, satellite services and Internet Service Providers (ISPs). In this redbook, we are interested in wireless service providers that provide Internet access. When developing mobile applications, it is very important to understand the wireless technologies supported by the wireless service provider in the targeted mobile user's area. The wireless service provider dictates the wireless network type, wireless protocol and thus the markup language used for application development, as well as the supported mobile device type. In addition, it is important to understand the specifications of the wireless service provider protocol gateway.

The protocol gateway of the wireless service provider, such as a WAP gateway, can vary in specification and support. For example, one service provider may support session control of mobile devices by its WAP gateway, and another provider may not. This type of consideration will have a significant impact on the design of your mobile commerce application and should be understood before entering the application development phase.

2.6 Next-generation technologies

In this section, we highlight some key technologies that will have a major impact within the mobile computing world in the near future. Most of these technologies already exist and are continually being updated or improved. The biggest question with these technologies is when will service providers and technology providers fully harness their capability and implement them for the mass market.

J2ME - Java 2 Platform, Micro Edition

The capability to run Java on a mobile device allows for many new possibilities that can be exploited.

Recognizing that one size doesn't fit all, Sun has regrouped its innovative Java technologies into three editions: Micro (J2ME technology), Standard (J2SE technology), and Enterprise (J2EE technology):

- Java Virtual Machines that fit the range of consumer devices
- A library of APIs, specialized for each type of device
- Tools for deployment and device configuration
- A profile or specification of the minimum set of APIs useful for a particular kind of consumer device (set-top, screenphone, wireless, car, and digital assistant) and a specification of the Java Virtual Machine functions required to support those APIs

Java 2 Platform, Micro Edition (J2ME) is a Java technology specifically designed to address the vast consumer market, which covers the range from extremely tiny commodities such as smart cards or pagers all the way up to the set-top box, an appliance almost as powerful as a computer. J2ME provides a Java Virtual Machine (JVM) capable of running on mobile devices.

J2ME platform maintains the qualities that Java technology:

- Built-in consistency across products of running anywhere, any time, over any device
- Portability of the code
- Same Java programming language
- Safe network delivery
- Applications written with J2ME technology are upwardly scalable to work with other Java editions

J2ME has been implemented by various technology providers, such as NTT DoCoMo's i-mode 503i. The 503i mobile phone includes J2ME support and is used by many Japanese mobile phone manufacturers.

Another example of a Java-enabled mobile phone is the Motorola i85s. This phone has been manufactured in partnership with Motorola Inc. and Nextel. The the i85s is among the first phones to use Java and comes loaded with a number of applications, including voice-activated dialing, a datebook, an enhanced phone book, a voice recorder, menu customization, and enhanced security.

More detailed information on J2ME can be found at:

`http://java.sun.com/j2me/`

Bluetooth

Bluetooth is a wireless personal area network (PAN) technology from the Bluetooth Special Interest Group, founded in 1998 by Ericsson, IBM, Intel, Nokia and Toshiba. Bluetooth is an open standard for short-range transmission of digital voice and data between mobile devices (mobile phones, wireless PDAs, wireless laptops) and desktop devices. It supports point-to-point and multipoint applications.

Bluetooth provides up to 720 kbps data transfer within a range of 10 meters, and up to 100 meters with a power boost. Unlike Infrared Data Association (IrDA),

which requires that devices be aimed at each other (line of sight), Bluetooth uses omnidirectional radio waves that can transmit through walls and other non-metal barriers. Bluetooth transmits in the unlicensed 2.4 GHz band and uses a frequency-hopping spread-spectrum technique that changes its signal 1600 times per second. If there is interference from other devices, the transmission does not stop, but its speed is downgraded.

More detailed information on Bluetooth can be found at:

`http://www.bluetooth.com`

High speed wireless networks

IMT-2000/UMTS will have a significant effect on the mobile computing and m-commerce market by increasing the bandwidth for mobile devices.

More detailed information on IMT-2000/UMTS can be found in 2.1.7, “IMT-2000” on page 39.

Voice technologies

Voice XML or VXML is an extension of XML that defines specific tags for voice and enables access to the Internet via telephones and other voice-activated devices. AT&T, Lucent, IBM and Motorola created the Voice XML forum to support this development.

More detailed information on Voice XML can be found at:

`http://www.vxml.org`

3

m-commerce development methodology

In this chapter, we provide the project manager, developer, architect and specialist with a development methodology for building m-commerce solutions. The information is structured with a focus on the unique issues related to mobile computing and m-commerce.

This chapter is organized into the following sections:

- Understanding WCS V5.1
- Market study of the customer environment
- Existing or new site
- Customer requirements
- m-commerce implementation approaches
- Application design considerations
- Application development
- Testing the m-commerce application

3.1 Understanding WCS V5.1

Before you start building m-commerce solutions using WCS V5.1, it is important to understand the systems architecture, programming model, and session management support in WCS V5.1.

3.1.1 WCS V5.1 systems architecture and programming model

We have provided some useful sources of information for understanding WebSphere Commerce Suite V5.1.

Systems architecture

Refer to:

- *Fundamentals, IBM WebSphere Commerce Suite V5.1*, provided on the product CD
- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167
- *WebSphere V3.5 Handbook*, SG24-6161

Programming model

Refer to:

- *Programmer's Guide, IBM WebSphere Commerce Suite V5.1*, provided on the product CD
- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167 (soon to be available in redbook format)
- *WebSphere Commerce Suite V5.1 Customization and Transition Guide*, SG24-6174 (soon to be available in redbook format)

3.1.2 Session management

WebSphere Commerce Suite V5.1 supports two different mechanisms for session management between the client device and the Web application:

- Cookies
- URL rewriting

Cookies

A *cookie* is a piece of information with a specific format that can be included in the HTTP headers. When the user visits the site for the first time, a new session ID for the visitor is generated and sent back to the browser through a cookie. The browser receives the cookie and stores it locally. If the user requests another page from the same site, the browser automatically includes the stored cookie before it is sent to the Web application.

Cookie-based session management in WebSphere Commerce Suite V5.1 uses two types of cookies:

- **Non-secure session cookie**

  This type of cookie is used to manage session data. It contains the session ID, negotiated language, current store and the shopper's preferred currency. This cookie can flow between the browser and the server on either an SSL or a non-SSL connection. There are two types of non-secure session cookies:

  – WebSphere Application Server session cookie (*sesessionid*)

    It is based on the servlet HTTP session standard and can be found in any document in the WebSphere Application Server. WebSphere Application Server cookies persist in memory or in the database in a multi-node deployment.

  – Commerce Suite session cookie (*wcs_session_id*)

    It is internal to Commerce Suite and is not persisted in the database.

- **Secure authentication cookie (*WCS_AUTHENTICATION_ID*)**

  This type of cookie is used to manage authentication data. An authentication cookie flows over SSL and is timestamped for maximum security. This is the cookie used to authenticate the user. Authentication cookies are always generated by Commerce Suite whenever cookie-based session management is in use.

Note: By default, both types of cookies expire when the user closes the browser.

Cookies are the most widespread means for session management, because they are transparent to the user and do not need any specific programming code. The problem is that many browsers either allow the user to disable the reception of cookies or, as in the case of some mobile devices (for example, i-mode phones), cannot handle cookies at all.

URL rewriting

As an alternative to cookies, WebSphere Commerce Suite V5.1 supports session management via URL rewriting, which consists of encoding the session ID as a parameter in any URL that the user can invoke from the current page.

For example, the link could be as follows:

```
<a href="store/catalog;$sesessionid$DA32242SSGE2">
```

Clicking this link causes the session ID to be sent in the HTTP request. URL rewriting requires some additional code to include these session IDs in the various links. For example, all links within a JSP will have to be rewritten as follows:

```
<%= response.encodeURL ("/store/catalog") %>
```

In addition to these changes, URL rewriting has a limitation: it limits the flow to generated pages only. In fact, static pages cannot be modified at runtime to include the session IDs. In this case, the user will be forced to visit only dynamic pages as long as a session is needed.

One major problem that is encountered when extending a Web-based application to mobile devices is their limited or missing support of cookies. Most mobile devices are not capable of storing cookies locally. In some cases, this limitation is overcome by giving a second entity the task of storing the cookies on behalf of the phone. For example, WAP gateways are also in charge of storing the cookies on behalf of WAP phones.

Note: Some mobile phones have limitations in the length of cookies. This may cause problems.

Some other technologies, for example i-mode, do not support cookies at all. One solution to this problem is to require session management from an i-mode specific adapter, or to use URL rewriting.

Note: Many mobile phones have limitations in the length of URLs. This should be taken into account when planning to use URL rewriting.

3.2 Market study of the customer environment

When developing applications for traditional e-commerce Web sites, one of the first tasks is to gather requirements. Of course, the customer requirements are still important, but before you make decisions on what is needed, you must understand the mobile device and wireless service provider constraints. For this reason, we recommend a market study of the customer environment.

Each geography has unique challenges with different types of mobile devices, number of providers, and different support offered by the providers. Such countries as Japan have one dominant device type or service, such as i-mode. In other countries, such as the US, there are many different possible devices and protocols to consider.

All mobile devices are not created equal. The size of the display and speed of the mobile network can dramatically affect the design of your application. It may not be appropriate or practical to develop a complex m-commerce full-function Web site if your target audience's mobile device type is a two-line display mobile phone. In this case it may be more practical to offer complementary m-commerce functionality to the user than to use the site with a PC browser. For example, a more limited m-commerce function may be to send an SMS message to the customer mobile device when the order is shipped. For mobile devices that offer more advanced functionality, it may be very desirable to create a full-service m-commerce Web site.

3.3 Existing or new site

What type of site does the customer want? There are several scenarios that you should consider when planning and designing an m-commerce Web site.

- Extending an existing e-commerce Web site for mobile clients

  It is important to understand if you need to extend an existing PC browser-accessible e-commerce Web site, and what the existing content is.

- New m-commerce Web site for mobile clients and PC browser clients

  When designing a commerce Web site from scratch, there are many design issues that could make the development of an m-commerce application more seamless in the way that the application is developed.

- New m-commerce Web site for mobile clients

  In some cases. the customer may only want a commerce Web site for mobile clients. The design implications for the runtime and development approach may be impacted by this decision, as we will discuss later.

3.4 Customer requirements

Before you perform a traditional customer requirements review, it is critical that you understand the limitations of the mobile infrastructure. This section will help you shape the requirements with the customer to fully realize what is possible.

3.5 m-commerce implementation approaches

There are two approaches for implementing m-commerce using WCS V5.1:

- **m-commerce direct**

  m-commerce direct is an implementation approach for supporting mobile devices directly connecting to the WCS V5.1 commerce Web site by detecting the mobile device and serving content with the appropriate markup language embedded in JSPs for the mobile device browser.

  For example, WAP mobile phones accessing an m-commerce direct enabled commerce Web site will be served content JSPs with WML markup.

- **m-commerce using WTP**

  m-commerce using WTP is an implementation approach for supporting mobile devices that use WebSphere Transcoding Publisher (WTP) as a filter to transcode WCS V5.1 content JSPs from either HTML or XML into the markup language supported by the detected mobile device browser.

  For example, an m-commerce application containing JSPs with simplified HTML output can be transcoded by WTP into WML for WAP mobile phones, and cHMTL for i-mode mobile phones.

There are some cases when a customer environment may require the use of both an m-commerce direct approach and the m-commerce using WTP approach. For example, you may want to provide optimal support for specific i-mode mobile phones, but also provide generic support for several other device sets, such as WAP mobile phones and PDAs.

3.5.1 m-commerce direct

The direct approach uses a device manager that receives a request containing information about a device from a servlet. It determines which adapter would best process the request, and passes the request to the appropriate adapter. Adapters are device-specific components that perform processing functions before passing a request to a controller.

Computing flow

1. To prevent applications from having to handle system functions, such as access control and authentication, requests from any device are first processed by the Commerce Suite Web Controller.
2. The adapter creates a session context and a controller request object, and passes the controller request object to the Web Controller.

The controller request object contains a set of properties, formatted by the adapter. It also contains a backward reference to the adapter object and a reference to the session context object created by the adapter.

3. The Web controller executes the request by invoking the corresponding controller command.

   All business logic is implemented in the controller command.

4. Based on the view name returned by the controller command, it will return the appropriate JSP file to the requesting device as seen in Figure 3-1.

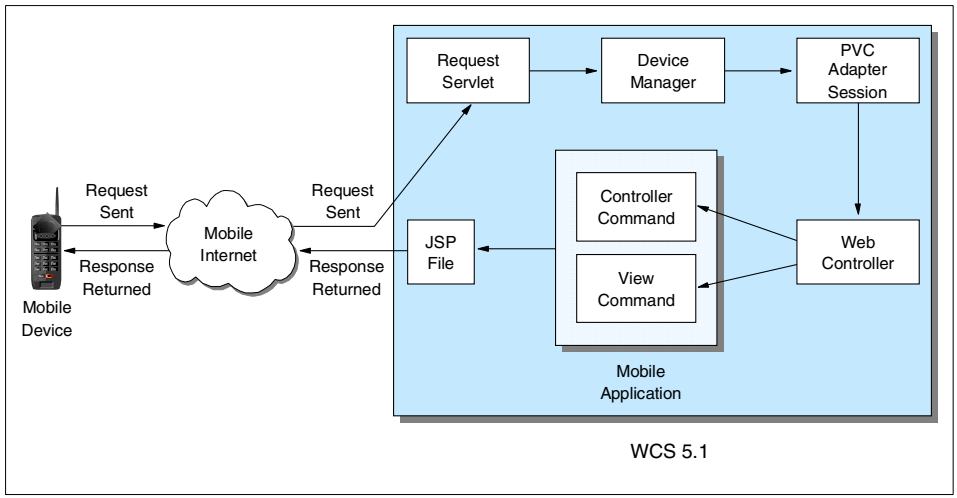


Figure 3-1 m-commerce direct - computing flow

To implement the m-commerce direct approach, you must configure the device manager to recognize the type of device that is assessing your store. This is accomplished by creating a PvC adapter and content JSPs with the appropriate markup language of the targeted mobile device.

The features and functionality added to WCS V5.1 to enable m-commerce can be found in Chapter 4, “m-commerce features and functionality in WCS V5.1” on page 85.

3.5.2 m-commerce using WTP - application runtime topology

One of the biggest challenges in mobile computing is providing support for the many different mobile devices, each potentially using its own unique presentation language, often referred to as a markup language (CHTML, WML, HDML, HTML, XML, and XHTML). Coupled with this, you need to be able to provide support in an efficient manner, taking into account low network bandwidths.

The WebSphere Transcoding Publisher transforms arbitrary content into a form that can be presented on a device different from the original target, such as changing HTML content intended for desktop PCs to WML content suitable for the new class of WAP mobile phones.

In general, transcoding is the process of transforming content from one format into another. This includes conversion between alternative screen or window sizes and aspect ratios, so that the content can be displayed on a wide and growing variety of devices. It also includes conversion into another language.

Both enterprise and Web content may be filtered, transformed, converted, or reformatted to enable them to be universally accessed by a variety of devices, to exploit specific application requirements for content customization, and to enable personalization of general content. Moreover, this content may be delivered over a wide range of networks, and as a result the network bandwidth and latency encountered will vary greatly. Figure 3-2 presents the computing flow of m-commerce using WTP.

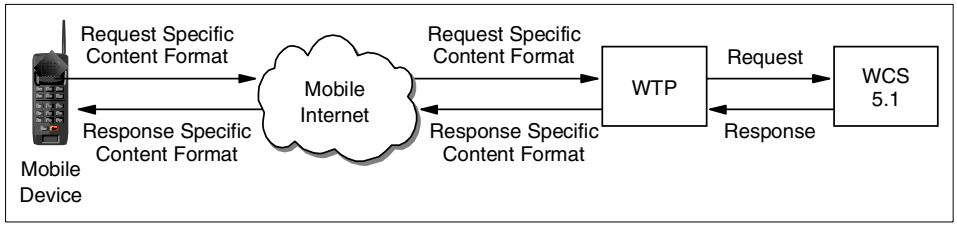


Figure 3-2 Computing flow of m-commerce using WTP

For more detailed information on WTP, refer to *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965.

3.5.3 Guidelines for selecting the implementation approach

This section provides considerations for selecting the m-commerce implementation approach. We have examined considerations based on functionality, extensibility, scalability, development effort and cost.

Key questions for selection input

Here are some questions to help in your selection:

1. What is the targeted mobile device type (for example, WAP mobile phones, i-mode mobile phones, etc.)?

2. How many different types of mobile devices exist within the category of mobile devices that you plan on supporting?

   For example, there are many manufacturers of WAP mobile phones. These mobile phones may have different specifications for screen size, memory, and URL buffering capability.

3. How many content JSPs will your application support?
4. Is your m-commerce application a full-function or limited-function commerce store?
5. Do you already have an existing commerce Web site with JSPs using HTML or XML content?
6. Does the m-commerce application need to be highly precise in the support and capability provided for the mobile device?

Pros and cons of m-commerce direct

Following are the pros and cons of the m-commerce direct approach as key decision points for making an implementation selection.

Pros:

- If you need a high level of customization and support for a specific mobile device, the m-commerce direct approach provides the most flexibility.
- If you have a limited number of mobile devices to support within a given mobile device category, m-commerce direct may be the best approach.

  For example, in Japan i-mode mobile phones are dominant. Developing JSPs with cHTML content may provide the best support for the targeted i-mode mobile phone.
- The acquisition cost of the direct approach is very low.

Cons:

- If you have many mobile devices that require separate JSPs with different markup languages, the direct approach may not be a good option because of the high cost of developing the content JSPs.
- Each device type needs a customized version of the JSP with specific content for the targeted mobile device.
- Development cost can be high if changes are required for more than one mobile device set of content JSPs.

Pros and cons of m-commerce using WTP

The theory behind using transcoding is that you need only put a transcoder in front of your existing commerce Web site. We have found that in practice there is a little more to it than that.

Following are the pros and cons of the m-commerce using the WTP approach as key decision points for making an implementation selection:

- You will need to modify your content JSPs to contain either simplified HTML or XML for output of WCS to WTP for transcoding.
- Transcoding offers advantages when support is needed for more than one mobile device category, with many variations within the device categories. Content JSPs can be developed using XML with style sheets for the targeted mobile device.
- The bottom line with transcoders is that they can be made to work, but must be tuned for identification, by the use of examples, or programmed to execute in the desired way by modifications. Any change in the back-end system is likely to require additional programming and training of the transcoder. The process of programming or training such a transcoder can be elaborate.
- Even with advanced transcoder engines, two problems still remain.
  - First, transcoders typically operate on an interaction-by-interaction basis and they are not usually designed to diverge from this mode. However, in order to adapt to characteristics of the device, it is necessary to modify the data flow across interactions. For example, if you have a store that requires seven mouse clicks to create an order, a transcoder supporting this service on a PDA device could, with some programming or training, translate the relevant HTML pages into a suitable format for the mobile PDA, but it would still deliver seven HTML pages that need to be navigated, which may not be desirable.
  - Second, transcoders usually support existing services. If the new devices require or permit new services or forms of interaction, transcoders have difficulty supporting them.
- In order to implement the WTP approach to m-commerce, you must purchase the WebSphere Transcoding Publisher.
- Using WTP, the maintenance cost for a simple store tends to be lower.

In summary, the m-commerce direct approach is recommended when you intend to generate content for one or maybe a few types of mobile devices and want a high level of customization. The m-commerce using WTP approach can be used when you need to transcode the WCS contents for many types of devices.

3.6 Application design considerations

Depending on the implementation approach selected, we have provided more detailed information on the application design considerations.

For more detailed information on m-commerce direct application design, refer to 10.1, “m-commerce direct application design guidelines” on page 200.

For more detailed information on m-commerce using WTP application design considerations, refer to Chapter 18, “m-commerce using WTP implementation and design” on page 295.

3.7 Application development

The application development of an m-commerce Web site will vary from the different implementation approaches mentioned above in the configuration of the runtime environment and content JSPs.

Following are the high-level steps for m-commerce application development:

1. Setting up your environment

   In order to develop an m-commerce application, you will need to set up the following:

   – Runtime environment

     The runtime environment will be used for unit and system testing your m-commerce Web site.

   – Existing commerce Web site

     If you have an existing store, you should publish or deploy the store to verify the runtime environment. An alternative to having your own store is to use the InFashion sample store.

   – Development environment

     In order to create the Java classes for the PvC adapter and content JSPs for the direct or WTP approachm you will need to install and configure the development environment and tools (Studio and VAJ).

   For more detailed information on setting up the runtime, development and test environments, refer to Part 2, “Setting up your m-commerce environment” on page 139.

2. Creating the PvC adapter

Once we know how to detect the mobile device, we can create the PvC adapter class in VisualAge for Java. We will implement three methods in the PvC adapter class: checkDeviceFormat, getDeviceModel, and getTerminalId.

For more detailed information, refer to 11.1, “Creating a PvC adapter” on page 206.

3. Deploying the PvC adapter

   Once we have developed the PvC adapter, we need to deploy the PvC adapter JAR file, update the **Command line argument** field in the WebSphere Advanced Administration Console. Next, update the WCS instance XML configuration file with the PvC adapter definition.

   For more detailed information, refer to 11.2, “Deploying a PvC adapter” on page 218.

4. Content management configuration

   Content management works with the PvC adapter to provide the proper JSPs for the detected mobile device. The content management configuration requires the WCS database tables to be updated to allow this process.

   For more detailed information, refer to 12.1, “Content management configuration” on page 226.

5. Creating device-specific content JSPs

   Whether you are developing an implementation approach using m-commerce direct or using WTP, you may want to modify the application flow. In WCS V5.1 JSPs are used to display contents of a Web page. For example, you may want to reduce the number of pages required for shopping, or possibly offer only reorder capability based on items in a quick list.

   If you are using the m-commerce direct approach, you will need to modify the content JSPs to include the markup language supported by the target device. For example, WAP mobile phones will require that JSPs be created with embedded WML content.

   If you are using the m-commerce using WTP approach, you will need to modify the content JSPs to include simplified HTML or XML with different XSL for each category of mobile devices.

   For more detailed information, refer to 12.2, “Create device-specific content JSPs” on page 232.

6. Deploying m-commerce content

   After you have developed your new JSPs, you need to publish them to a unique directory within the application target directories of your runtime environment. For example, if your commerce Web site supports PC browser clients, you may have a directory structure that looks something like the following:

Example 3-1 Example directory structure for m-commerce content

```
\stores\InFashion\*.jsp (PC client JSPs)
\stores\InFashion\wml_jsp\*.jsp (WAP WML content JSPs)
or
\stores\InFashion\wtp_xml\*.jsp (WTP XML content JSPs)
```

   Note: One of the functions of the PvC adapter is to detect the mobile device type and set the correct directory to load JSPs with the appropriate content for that mobile device.

7. Creating custom commands (optional)

   For information on this topic, refer to Chapter 13, “Creating custom PvC commands” on page 255.

3.8 Testing the m-commerce application

When developing an m-commerce application, there are several test environments that need to be implemented to ensure proper testing is done prior to production deployment.

When developing the m-commerce application within VisualAge for Java and WebSphere Studio we recommend that you set up the following configuration to test and debug your application:

- WebSphere Test Environment with WCS runtime

  The WebSphere Test Environment (WTE) is included in VisualAge for Java (VAJ) for debugging WebSphere Application Server applications. WebSphere Commerce Suite V5.1 provides a VAJ repository file to import the WCS environment into WTE, to run WCS application code and have full debug capability. This environment is used in application development and unit testing.

  For more detailed information on m-commerce development using the WebSphere Test Environment, refer to *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167.

- WebSphere Commerce Suite V5.1 runtime

  In order to fully test your m-commerce application with the PvC adapter, you will need to set up a WCS V5.1 runtime environment.

  For more detailed information on m-commerce runtime environments, refer to Chapter 7, “m-commerce runtime environment” on page 141.

- Simulator testing

  In the early stages of m-commerce application development it is highly desirable and cost effective to use software simulators for the mobile device client testing. The simulators are provided by various manufacturers. Several manufacturers provide a software developer’s toolkit (SDK) containing documentation, a simulator, and sample code.

- Real hardware

  As your testing progresses, we recommend that you build an environment that supports real wireless hardware devices. There are many variations that can be implemented. We recommend that you provide the following two environments for real hardware testing:

  – Real wireless hardware - intranet test environment

    Before deploying the m-commerce application code to production, it is required that you test with real mobile devices. The intranet test environment allows you to perform nearly all testing without having to have your application code outside your company’s firewall.

  – Real wireless hardware - Internet test environment

    In final preparation for production, or if you are developing a m-commerce application for a wireless protocol that requires services from a wireless service provider you will need to set up an Internet test environment.

For more detailed information on m-commerce test environments, refer to Chapter 8, “m-commerce development environment” on page 159. In addition, information specific to the mobile device type is provided in the following implementation sections specific to a wireless protocols:

- Chapter 14, “HDML implementation sample” on page 265
- Chapter 15, “WAP implementation sample” on page 271
- Chapter 16, “Palm implementation sample” on page 279

4

m-commerce features and functionality in WCS V5.1

Along with the many advantages and business opportunities that m-commerce presents, there are some technology challenges in properly supporting mobile devices for m-commerce. WebSphere Commerce Suite V5.1 is ideal for m-commerce because of its mobile device support features. These m-commerce features are known as PvC (pervasive computing) extensions.

The chapter is organized into the following sections:

- WCS V5.1 m-commerce enablement overview
- WCS V5.1 m-commerce enablement features
- Session control
- Device control
- Security
- URL buffering

4.1 WCS V5.1 m-commerce enablement overview

This section describes the unique challenges presented by mobile devices, as well as a functionality overview of the features added to enable WCS V5.1 for mobile commerce.

Session control

Session control provides a means for a Web server to keep track of the client system users' transactions, valid URLs and preferences. On PC browser clients, session control is often maintained by using a cookie in cooperation with the browser, such as Microsoft Internet Explorer or Netscape Navigator.

In the mobile computing world, some mobile devices do not support cookies on the client device. In other cases, the session is maintained by the service provider protocol gateway; such is the case for WAP.

For this reason, the Web server needs another way to identify such a cookie-less device as a client.

WCS V5.1 supports a function that enables session control between a Web Server and mobile device by using a unique identifier stored in a WCS PvC session table called PVCSESSION. The unique identifier information is sent in the HTTP request header provided by the mobile telecommunication carrier.

Device control

There are several categories of mobile devices such as mobile phones, PDAs, and wireless laptops. Within the categories of mobile devices, there are often unique characteristics and specifications. For example, two WAP phones, whether from the same manufacturer or different manufacturers, may vary from each other in the following features:

- Microbrowser
- Size of screen
- Screen resolution
- Color capability
- Size of memory

WCS V5.1 provides the ability to detect the mobile device type via the incoming HTTP request from the mobile device, and send back the JSPs with the appropriate content and images for that mobile device. This requires content JSPs and images with the appropriate markup language and graphics to support the target mobile device.

Security enhancement

A mobile device is easily lost or stolen because it is generally small. Also, in light of the portability of mobile devices, the chances of your user ID and password being seen in a public space increase. To address these security issues, WCS V5.1 includes enhanced security functions for logon timeout, command execution restriction, and user registration control.

URL buffering

Generally speaking, there is a limitation in the length of HTTP requests that can be transmitted from a small device such as a portable cell phone. Lengthy URLs with parameter contents, for example the registration input form for a user profile, may be cut in half due to the small memory in the device. To resolve this problem, WCS V5.1 provides URL buffering functions. By using the PVCBufferUrl function, we can divide the original content (perhaps content originally designed for PC browsers) into several pieces, and send these pieces to the WCS Server separately. The WCS Server temporarily buffers the separate data in the WCS database. After all data is sent, the command is executed using the buffered data.

4.2 WCS V5.1 m-commerce enablement features

WebSphere Commerce Suite V5.1 includes the following features to enable m-commerce:

- PvC adapter framework
- PvC commands
- PvC data beans

4.2.1 PvC adapter framework

The PvC adapter framework includes the following components:

- PvCAdapterImpl class
  - checkDeviceFormat method
  - getDeviceModel method
  - getTerminalId method
- PvC adapter definition for WCS instance XML configuration file
- PvC adapter WCS database table updates

PvCAdapterImpl class

A PvC adapter is implemented as a Java class that extends the class com.ibm.commerce.pvcadapter.PVCAdapterImpl as the super class. The PvC functionality is included in the PvC adapter framework included in the WCS V5.1 class libraries.

The values returned by the methods implemented in the class depend on the wireless protocol and gateway specification. Each adapter class is required to implement these methods to provide carrier-specific information for the adapter.

When developing a PvC adapter for your target mobile device type, the following methods must be used:

- checkDeviceFormat

  This method checks if the necessary information for session control is contained in the HTTP request from the browser.

- getDeviceModel

  This method is used to get the mobile device model name from the HTTP request.

- getTerminalId

  This method is used to get the unique ID from the mobile device HTTP request.

Important: If your mobile client type or wireless service provider does not supply a unique ID, the WebSphere Application Server (WAS) session information can be used. For example, we used the WAS cookie as the unique identifier.

We explain how to implement these methods in 11.1, “Creating a PvC adapter” on page 206.

Deploy the PvC adapter

This section describes how to deploy a PvC adapter.

Deploy the PvC adapter JAR

The PvC adapter code will need to be exported from VisualAge for Java to a JAR file and then copied to the <wcs_install_path>\lib directory. Next, the JAR will need to be added to the command line arguments of the WebSphere Commerce Server from the WebSphere administrator’s console.

PvC adapter definition

In order to deploy the PvC adapter, entries need to be added to the WCS instance XML configuration file. The WCS instance entry attributes, possible values, and usage example can be found in “PvC adapter definition for the WCS instance XML file” on page 330.

Content management configuration

This section describes the minimum configuration necessary to provide the desired content JSPs for the detected device type of the PvC adapter. The following WCS database tables should be configured:

- PVCDEVMDL

  This table stores model information about PvC devices.

- PVCDEVSPEC

  This table stores specification information and the JSP content directory. One record can be shared by more than two devices.

- PVCMDLSPC

  This table stores the relationship between the model and the specification. Models of devices that have similar specifications are categorized to the same record in the PVCDEVSPEC table.

The following list explains the workings of content management:

1. When a request comes from the browser, the server extracts the model name from the PvC adapter.
2. The server then looks for the record related to the extracted device model and adapter name.
3. If a suitable record for the client’s device is found in the PVCDEVMDL table, which is determined by the relationship stored in PVCMDLSPEC table, a suitable device specification can be found in PVCMDLSPEC, and the server will use the value of PVCMDLSPEC.CONTENTDIR for the selected record for content switching.
4. If there is no model-spec relationship in the PVCMDLSPEC table, the server will use the adapter default model-spec relationship to select a suitable record in PVCMDLSPEC for the client's device.

Note: If you do not need to change the JSP root directory for the newly created PvC adapter, and you do not need to use data stored in the PVCDEVSPEC table, you do not need to set up the database to manage your content.

For example, if you are developing a store that only supports mobile devices of the same type, you can modify the JSPs in the default directory of the VIEWREG table.

For more detailed information on syntax and usage, refer to “Content management reference” on page 364.

PvC adapter framework reference

For details about the syntax and usage, refer to Appendix A, “PvC adapter framework reference” on page 329.

4.2.2 PvC commands

The PvC commands are used within the content JSPs for merchants who wish to manage user and device information.

PVCRegistration

This command enables registration and renewal of the users’ PvC device information. Registration records of users and PvC device information records can be made and updated. This command is used together with the Secure Socket Layer (SSL) to encode the user's logon ID, password and individual information.

PVCRegistrationDevice

This command enables registration and update of PvC information of users that have already registered. It creates and updates PvC device information. This command is used together with the Secure Socket Layer (SSL) to encode the user's logon ID, password and individual information.

PVCChangeDevice

Some wireless protocols and service providers require a one-to-one relationship between the device and user. This is offered as a security precaution. The PVCChangeDevice can be used by updating the PvC adapter definition `registrationMode` value to `2` in the WCS instance XML configuration file.

You need to manage canceled or changed devices, because once a user has registered a device, no further registration is allowed using another device for the account in the WCS database until the old user-device relationship is canceled.

PVCBufferUrl

This command enables buffering of input field data placed in multiple pages, and delivers this data to one command. It saves parameters of the destination URL on the server in a temporary database table. Buffers that have not been updated for some time will be inaccessible. The time until the buffer becomes ineffective is specified (in minutes) as bufferTimeout in the configuration file.

ReEnterPassword

This command adds the given ReEnter Password parameter to the specified URL and redirects. Usually the ReEnterPassword command is used by a JSP that is assigned to ReEnterPassowrdForm. ReEnterPasswordForm is called when executing a command without a password in password-locked status.

PvC command reference

For detailed information on syntax and usage, refer to Appendix B, “PvC command reference” on page 339.

4.2.3 PvC data beans

The PvC data beans provide access to buffered parameters and the user device address.

Note: The PvC data beans packaged in the additional material zip file associated with this redbook are not officially supported by IBM.

PVCBufferDataBean

This data bean is required if you want to write a JSP page with parameter buffering.

UserPVCDeviceDataBean

This data bean allows you to access a user’s device address stored in the USERPVCDEV table. You can use this data bean when you need to extract data such as an e-mail address from the USERPVCDEV table.

PvC data beans reference

For detailed information on syntax and usage, refer to Appendix C, “PvC data bean reference” on page 355.

4.3 Session control

This section describes the session control functionality provided in WCS V5.1 for m-commerce enablement.

4.3.1 Unique identifier

The unique identifier in the HTTP header request is used to enable session control within WCS V5.1. We can identify mobile devices that do not support cookies, such as a mobile phone, by using this capability.

In some cases the unique identifier is issued by the gateway of the wireless service provider or the telecommunication carrier that provides the Internet connection service to the mobile phone user, as seen in Figure 4-1. The uniqueness of this identifier is assured by the carrier. The identifier is usually associated with the mobile phone number.

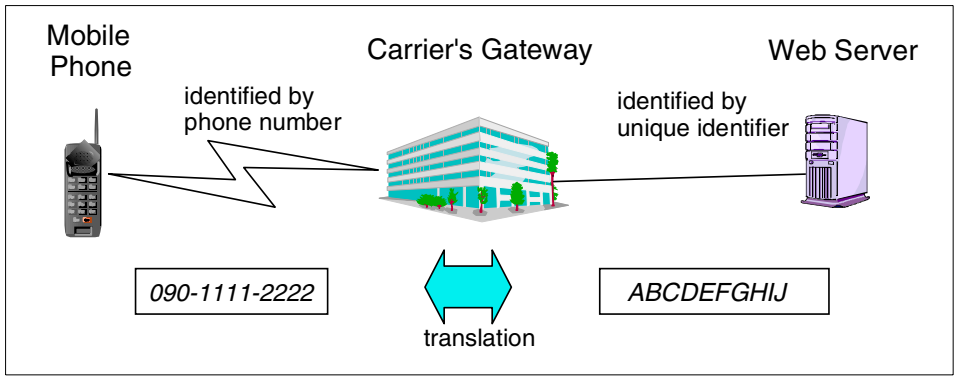


Figure 4-1 Unique identifier flow

4.3.2 PvC adapter

The carrier gateway pads the unique identifier in the HTTP header request sent to the Web server. Different carriers implement padding of the unique identifier differently. WCS V5.1 includes a PvC adapter framework to create a PvC adapter. This PvC adapter extracts the unique identifier from the HTTP request sent in different formats by the carriers.

Note: This means one adapter is needed for each carrier.

The identifiers sent from different carriers might be the same if each carrier implemented its identifier in its own way. To avoid the possibility that the identifier from the carrier is not unique, WCS5.1 combines the identifier with the PvC adapter identifier that is generated by WCS V5.1. The combination of the carrier identifier and the PvC adapter identifier makes all mobile devices unique for all carriers, as depicted in Figure 4-2.

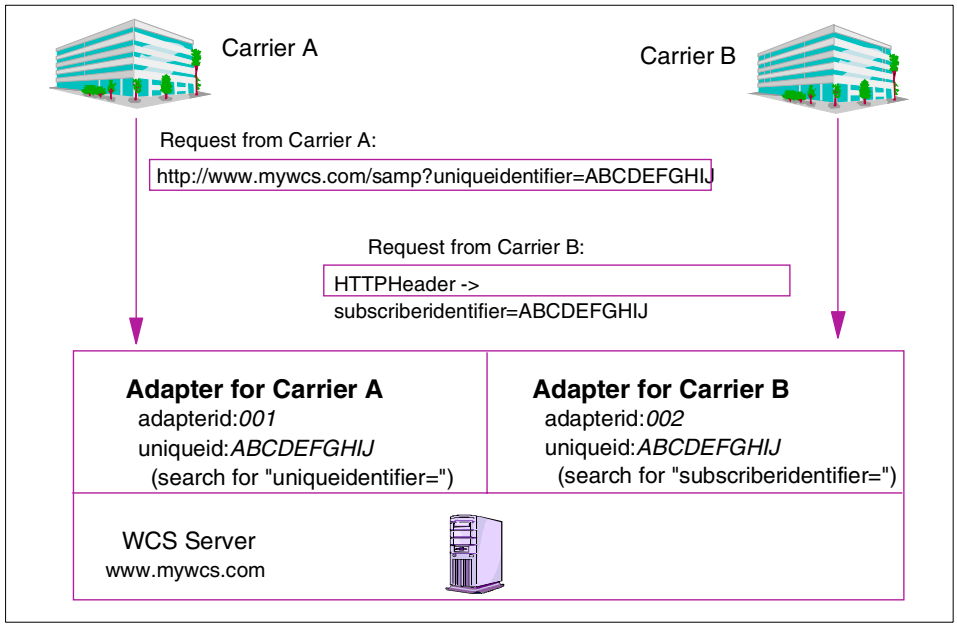


Figure 4-2 PvC adapter - unique identifier

The adapter ID is stored in the sessiontype field, and the unique identifier is stored in the terminal field in the WCS database PVCSESSION.

The advantage of the adapter is that we can separate the session control logic from the business logic.

Note: Adapters for different carriers or wireless protocols are not included in WCS V5.1.

- The PvC adapter framework is provided to create your own PvC adapter for your specific needs. The fundamental framework for creating a PvC adapter has already been prepared as Java classes and can be developed using 10 or 20 lines of code. We explain in detail how to create a PvC adapter in Chapter 11, “Creating and deploying a PvC adapter” on page 205.
- In addition, we provide sample code for PvC adapters in the sample implementation chapters including:
  - HDML: Chapter 14, “HDML implementation sample” on page 265
  - WAP: Chapter 15, “WAP implementation sample” on page 271
  - WTP: 19.5, “Creating a generic PvC adapter” on page 316

4.4 Device control

This section describes the functionality provided in WCS V5.1 for device control.

4.4.1 Differences between mobile devices

The differences between mobile devices include the unique identifier, hardware specification, wireless protocol, and the browser.

Markup language

Content is the information presented to the user on a Web page. A markup languages is a set of labels used to format individual elements, or groups of elements for the display or rendering of content on Web pages. HTML is the most widely used markup language for Web pages. Each mobile device has an underlying protocol that it supports. The protocol and browser determine the type of markup language that the device supports.

For example, WAP mobile devices widely in use today in Europe use the WML markup language. In Japan, the three major carriers have different implementations: compact HTML for i-mode, HDML for EZweb, and MML for J-SKY. The image file format is also different, as seen in Table 4-1.

Table 4-1 Three major carriers in Japan

Service Name	i-mode	EZweb	J-SKY
Markup Language	compact HTML	HDML (1)	MML(cHTML)
Image File Format	GIF	BMP, PNG	PNG
Max data transfer size (2)	2 KB	1.5 KB	6 KB
(1) Handheld Device Markup Language: almost compatible with WAP (2) recommended value by each carrier over device dependencies			

To support mobile devices using the m-commerce direct approach, developers will need to prepare different content JSPs with the appropriate markup language and image format for the targeted device.

Browser

The browser or microbrowser on a mobile device varies with different manufacturers and wireless protocols. Some device vendors implement their own browser and others use a vendor-developed browser. For example, OpenWave Phone.com developed a WAP-compliant browser used by Motorola, and Microsoft developed the Mobile Explorer used by Mitsubishi. The CompactNetFront* Browser is widely used not only on cell phones but also in PDAs, computer games, TVs and for car navigation in Japan, as seen in Table 4-2.

Table 4-2 Browser difference of i-mode terminal in Japan

Hardware Vendors	Browser
Mitsubishi, Fujitsu, NEC (N299i/F209i/D209i/N50ii/F501i/D501i/N502i/F502i/D502)	CompactNetFront (1)
Panasonic	proprietary
Sharp	proprietary
Note: (1) Copyright of Access Corp. (`http://www.access.co.jp/`)	

Display - color, size and resolution

Some devices have monochromatic screens and others are in 256, 4096, or 65536 colors. The size of the screen, number of lines of text, and resolution are also different. Table 4-3 provides a list of the differences in screen size by different i-mode mobile phones.

Table 4-3 The difference in screen size among i-mode devices (V-code 7316)

Device Model	rows x columns	number of characters on screen
P503, N502, N501	20 x 10	200
F503	16 x 10	160
SO503	20 x 9	180
P501, P502	16 x 9	144
NM502	16 x 8	128
F502, D502	16 x 7	112
F501, D501	16 x 6	96

Even if it is possible to see a gray scale bitmap image on a color screen, the merchants of an m-commerce site will want to customize the content and images to display their items as well as they on a mobile device. The page design and maximum number of lines available on a small screen need to be considered.

Other differences

Some new mobile devices have Java support (J2ME), some have a large amount of memory, some have an embedded digital camera, and others act as portable headphone stereo recorders. The merchant should customize the content of the JSPs and images to best support the mobile device.

4.4.2 WCS V5.1 functionality for device differences

By using the PvC adapter in WCS V5.1, mobile devices can be detected by using the device information sent by the carrier in the HTTP header. Once the device is detected, the appropriate JSP content and images are used for the mobile device. Using the appropriate image format supported by the wireless protocol of the mobile device can provide a more customized and attractive Web page for the mobile device shopper, as seen in Figure 4-3.

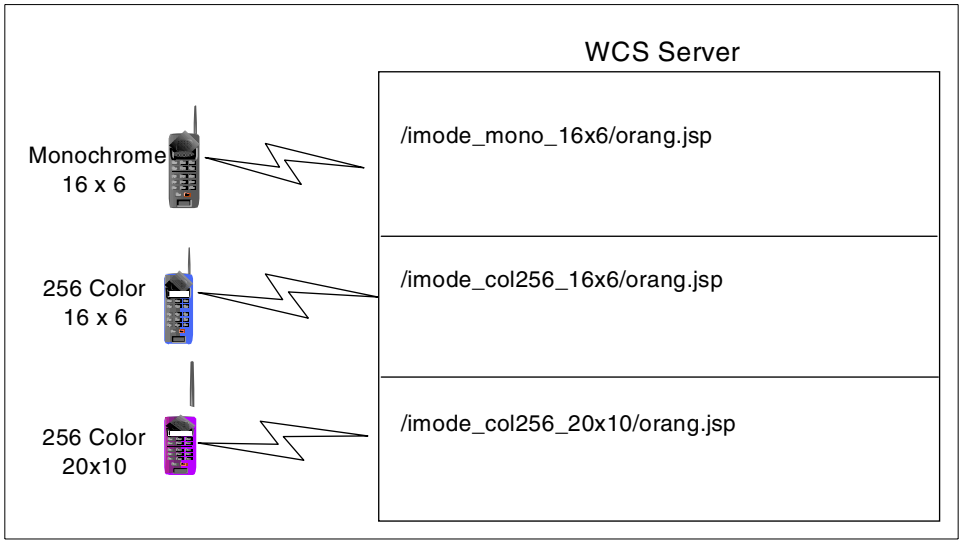


Figure 4-3 Device handling by WCS V5.1

4.5 Security

This section describes the functionality added for m-commerce security.

4.5.1 Logon timeout

A mobile device is small and could easily be lost or stolen. If you lose your mobile phone while you are logged on to a shopping site, the person who finds or steals your mobile phone can potentially purchase anything they want at your expense. For this reason, the user of a mobile device is at higher risk than a PC browser client user when shopping on a commerce Web site.

To provide enhanced security for mobile device users, WCS V5.1 has a logon timeout function for mobile devices. The site administrator can set the session timeout value (in minutes) in the PvC adapter section of the WCS instance XML configuration file. Details on these settings can be found in “PvC adapter definition for the WCS instance XML file” on page 330.

When there is no access within the time period specified by the timeout value, WCS V5.1 will automatically lock the session. After the timeout, you can no longer access the site until you enter your password again when prompted, as seen in Figure 4-4. WCS5.1 will restore the expected contents that the URL has stored in the database, and you can continue shopping again.

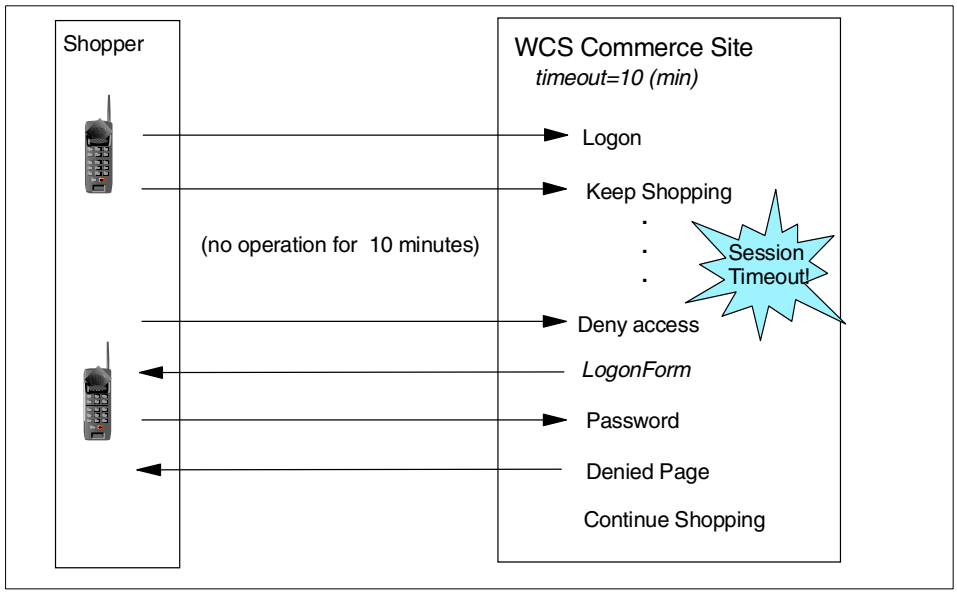


Figure 4-4 Logon timeout

4.5.2 Restricted command execution

WCS V5.1 can set up two types of restrictions for executing commands: one is the password protection mode, and the other is the unexecutable mode. These settings can be modified by editing the WCS instance XML configuration file, and the definition is valid only for a certain PvC adapter.

Password protection mode

We can choose the password protection mode for commands such as OrderItemDisplay, OrderDisplay, OrderProcess, or your own custom-designed commands, which need to be more secure than other commands.

Once the commands are set up in the password protection mode, users have to enter their password to execute the commands from the mobile device. Even if the mobile device is stolen, the thief will not be able to buy anything or access the owner's personal information without a password.

Unexecutable mode

This setting is useful if we encounter a security problem for a certain mobile device or carrier's service. We can stop the commerce service by setting up commands in the unexecutable mode.

4.5.3 User registration mode

For the users who have already registered with a commerce site from a PC browser client, it is useful to be able to log on to that site from a mobile device using the same user ID and password. Most shoppers who want to by something using a mobile device probably have already purchased something by using their PC. It is much easier for them to put in their home address or e-mail address by using PC keyboard. On the other hand, the commerce site should provide the ability to enter additional information for the mobile user, such as a mobile device e-mail address or a mobile phone number. This functionality gives the merchant the ability to send information or notification to the mobile device directly.

WCS V5.1 provides three types of user registration modes for the m-commerce site. In any case, the single user ID and password can be used between the mobile device and the PC browser client.

Normal mode (default)

If users have registered with a PC browser client, they can also log on to the site from a mobile device without registering again, as seen in Figure 4-5.

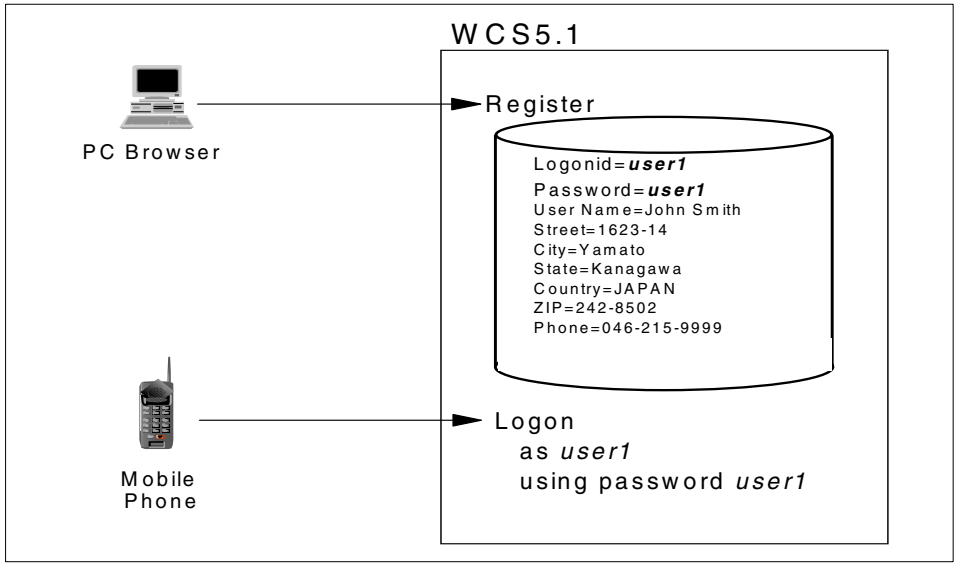


Figure 4-5 Normal mode for user registration

Register once

At the first attempt to log on to the Web site from a mobile device, WCS5.1 will send a new registration form that lets users input their mobile information, such as an e-mail address, for their own mobile devices, as seen in Figure 4-6. They can still use the previous user ID and password, and they do not need to register their personal information again. This information is stored in the WCS database.

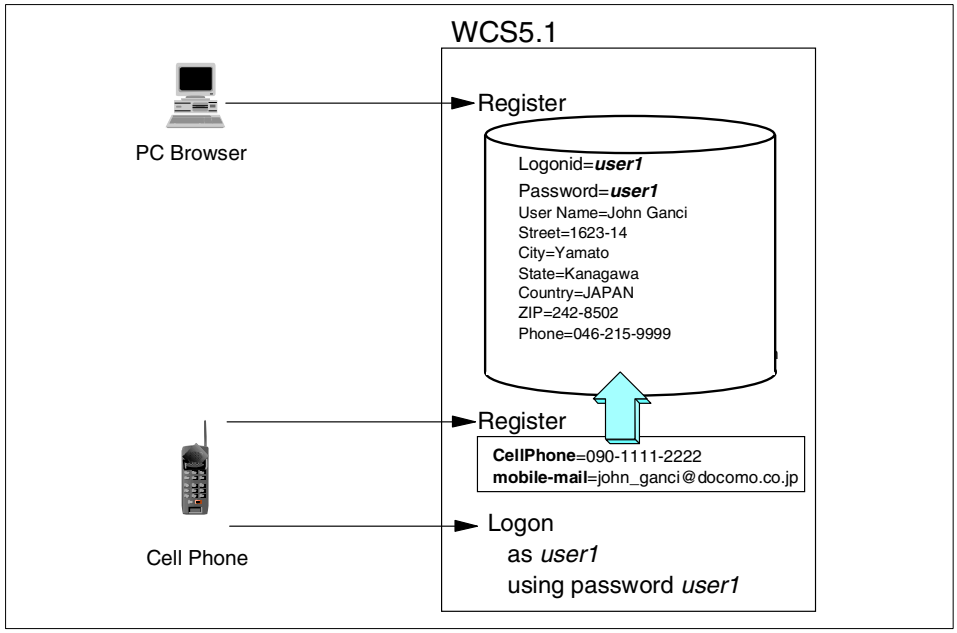


Figure 4-6 Register once

Restricted registration

This mode is similar to the register once mode, but more restrictive. Only one user can be registered to the commerce site for a certain mobile device, which means other people cannot use this mobile device to access that commerce site. Moreover, this user cannot access the same commerce site from a different mobile device using the same user ID. This means the relationship between a user ID and his/her mobile device becomes one-to-one as seen in Figure 4-7.

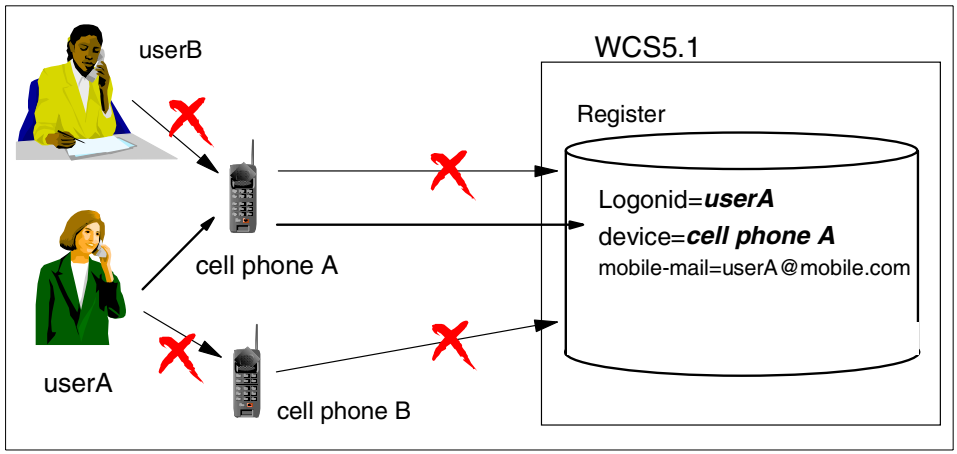


Figure 4-7 Restricted registration mode

As a result, the content provider can recognize who is accessing the site by checking the mobile device unique identifier. The content provider can then send a customized message tailored to the user's personal interests, such as the following:

Example 4-1 Restricted registration mode example message

```
Welcome John! How is the fishing?
(Please Enter your password)"
```

In this case the user does not need to enter even his/her user ID. This mode enables one-to-one marketing for the merchant.

4.6 URL buffering

Generally speaking, there is a limit on the maximum length of a URL that can be sent from a mobile phone. This limit depends on the memory size of the mobile device. For i-mode devices, we recommend that the length of a URL be less than 300 bytes. If we send a long URL, for example the registration form of the user profile, the URL might get cut in a half. To avoid this problem, WCS V5.1 provides the URL buffering function, as seen Figure 4-8.

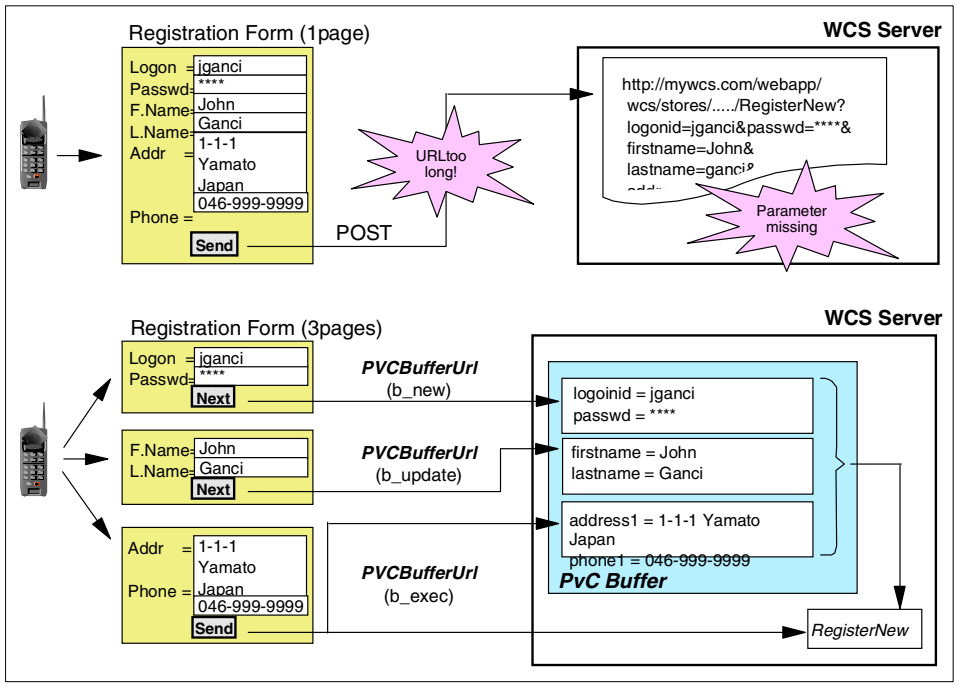


Figure 4-8 PVCBufferUrl

If we have a long URL that was originally made for PC browsers, we have to divide it into several pieces to suit the mobile device. There may be some input fields in each piece. The data will be stored in the PvC Buffer of the WCS server by using the PVCBufferUrl command.

The PVCBufferUrl command has three modes, as follows:

- b_new
- b_update
- b_exec

4.6.1 PVCBufferUrl command b_new mode

The b_new mode of URL buffering creates a buffer in the WCS database and stores the parameters in the first page. Figure 4-9 shows that when the user types in the value of data A, B and C and clicks the **Next** button, the PVCBufferUrl command stores this data (that is, A=1, B=2 and C=3) into the PvC Buffer in the WCS server, and WCS server will send the next “Page2” to the client.

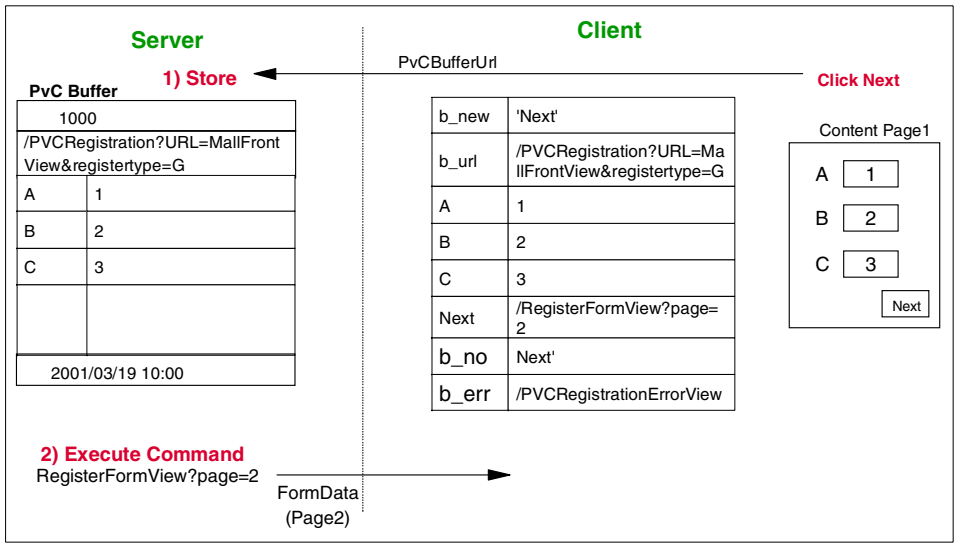


Figure 4-9 PVCBufferUrl b_new

4.6.2 PVCBufferUrl command b_update mode

On page 2, the user types in the value of data D, E and F and clicks Next. The PVCBufferUrl command then updates the PVCBuffer by inserting this new data. Next the WCS server sends the next page (page 3) to the client, as seen in Figure 4-10.

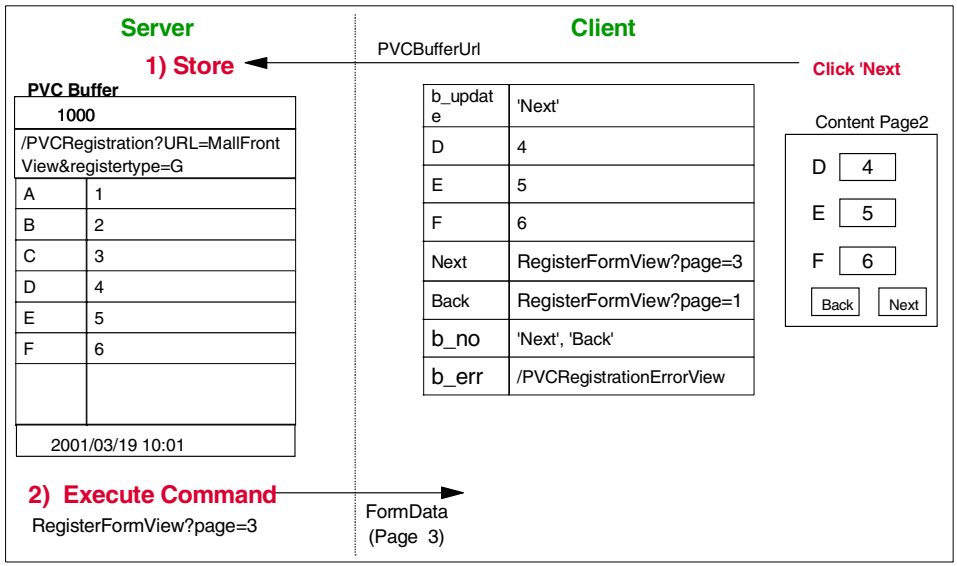


Figure 4-10 PVCBufferUrl b_update

4.6.3 PVCBufferUrl command b_exec mode

On page 3, the user types in the value of data G, and pushes Send. The PVCBufferUrl command then inserts this data into the PvC Buffer. Next, the WCS server executes the PVCRegistration command with all buffered data (A-G), and sends a form with the data to the client, as seen in Figure 4-11.

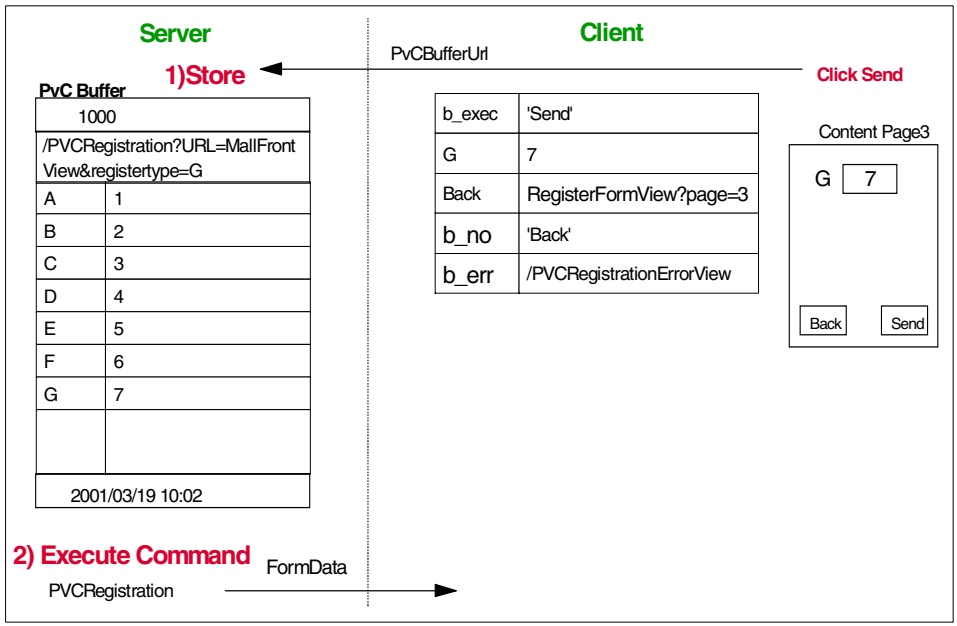


Figure 4-11 PVCBufferUrl b_exec

5

IBM wireless middleware

In this chapter, we discuss how WebSphere Commerce Suite V5.1 can be used with other key IBM wireless middleware products to build an m-commerce solution. We provide an overview of each product and cover considerations when deploying and integrating the products.

The chapter is organized into the following sections:

- IBM WebSphere Everyplace Suite (WES)
- IBM WebSphere Transcoding Publisher (WTP)
- IBM Everyplace Wireless Gateway (EWG)
- IBM MQSeries Everyplace (MQe)
- IBM DB2 Everyplace
- IBM Mobile Connect

5.1 IBM WebSphere Everyplace Suite (WES)

In today's fast-moving business world, it is important to keep up with the ever-changing demands of users. Computing is no longer confined to desktop PCs and people now expect to be able to access information at anytime and from any place with the device they happen to be using at the time. Accordingly, the applications in this environment are varied and encompass everything from simple Web browsing to secure access to business-critical data. There are, therefore, many challenges to be addressed in this environment, and the WebSphere Everyplace Suite contains a set of products that address these issues.

5.1.1 WES overview

IBM WebSphere Everyplace Suite V1.1.3 provides the functionality necessary to enable both network access and application and content serving to multiple device types. It also provides the functionality to extend e-business applications to the new classes of pervasive computing devices discussed previously, including WAP phones, PDAs, Internet appliances, and screen phones, in addition to the large base of Internet browsers.

Everyplace Suite is not part of the infrastructure that comprises those business applications and data, but rather it provides the infrastructure that allow many different devices to access those applications and data via networks. This is illustrated in Figure 5-1, which shows how WES logically fits into a deployment of a multi-channel, end-to-end e-business solution.

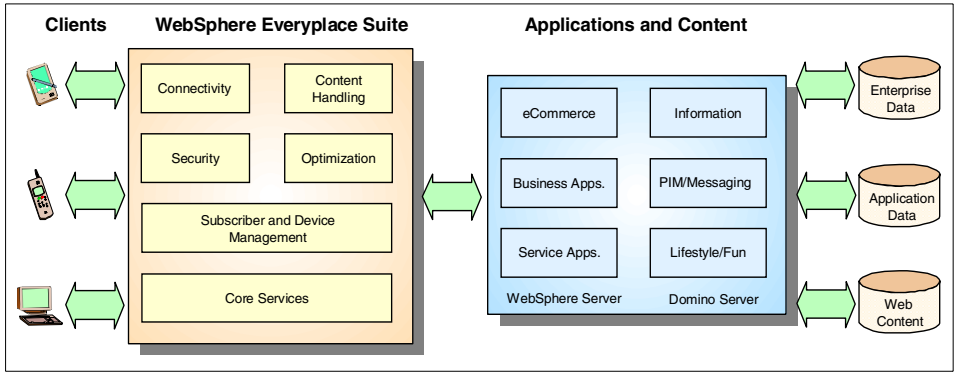


Figure 5-1 WES deployment

From a functional perspective, the IBM WebSphere Everyplace Suite is an integrated, modular suite of software components. Together, the Everyplace Suite components provide solutions for connectivity, security, content handling, optimization, and subscriber and device management.

Components

Each component performs a different function in extending pervasive computing connectivity. Figure 5-2 shows the product components that comprise the WebSphere Everyplace Suite under the function they perform.

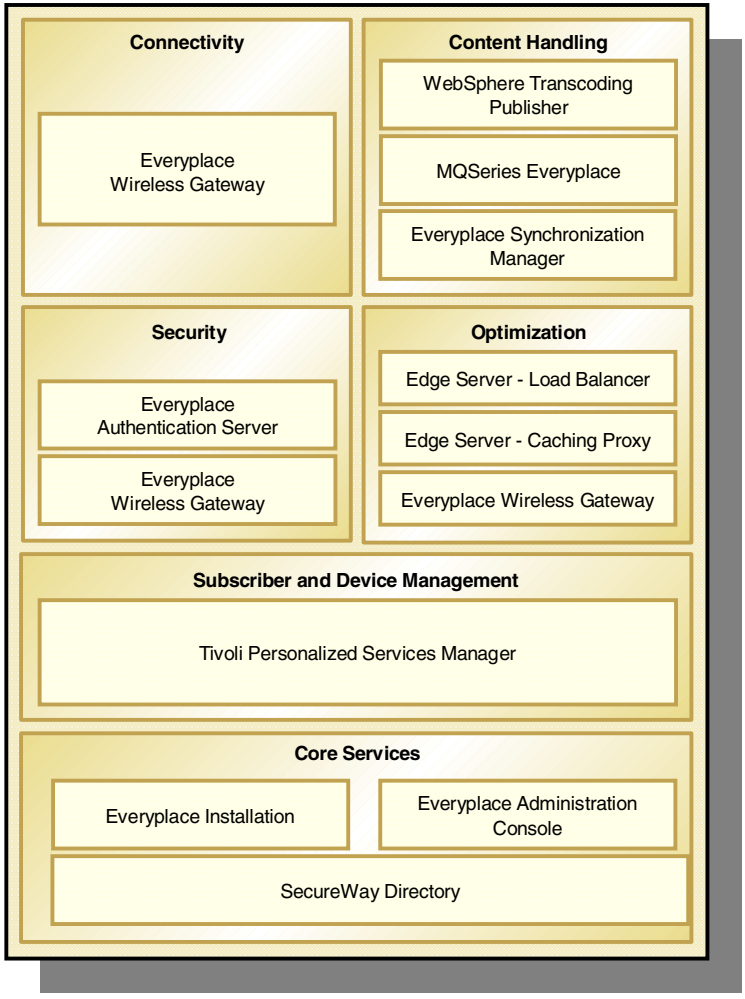


Figure 5-2 WebSphere Everyplace Suite functions and components

Connectivity

This function relates to how devices physically connect to your network. This is composed of two parts. First of all, you need gateways to provide the access from different types of wireless and wireline networks. Secondly, you need a protocol converter to take data transmitted over a specific network type and convert it into a form that can be used by an application.

IBM Everyplace Wireless Gateway provides secure wired and wireless connectivity between your enterprise network and whatever external communications networks that you need to support, for example, GSM, CDMA, TDMA, X.25, etc. It also implements protocol translation, and provides support for interfacing to short messaging centers via a series of APIs.

Security

IBM WebSphere Everyplace Suite provides an integrated security model that gives the user a significantly enhanced end-user experience. The key feature is the ability to provide single sign-on across all the components of an end-to-end e-business solution, which fully integrates with standard security techniques such as virtual private network (VPN) technology, firewall protection, etc.

IBM Everyplace Authentication Server forms the core of Everyplace Suite security functionality. It provides user and device authentication capabilities that together enable a single, device-independent user sign-on. It also provides the pass-through of authentication information to business application and data servers.

IBM Everyplace Wireless Gateway provides Virtual Private Network (VPN) support, which allows an enterprise to extend its private intranet across a public network, such as the Internet, creating a secure private connection by way of a private IP tunnel.

Content handling

Content handling is the process of moving content from your e-business solutions (either applications or data) to the point where it is required. This typically has two main aspects; moving the content and then ensuring that when the data is at the point where it is needed, it is in a form that can be used. To that end, this process comprises three main areas:

- Transcoding

  Since pervasive computing devices come in all sizes and have widely varying display capabilities, transmitted content must be customized to fit the capabilities of the requesting device.

- Asynchronous messaging

  Asynchronous messaging is a complementary technology to Internet browsing that is designed for applications where a “fatter” client is acceptable, and where disconnected modes of operation are required. To support these modes, it provides assured, asynchronous, once-only delivery of data across a broad range of hardware and software platforms.

- Data Synchronization

  IBM WebSphere Everyplace Suite has the capability to manage the automatic exchange and updating of e-mail, schedules, transactions, and database exchanges between popular pervasive devices and database servers. This allows users to perform work offline, and connect to the network whenever it is convenient.

IBM WebSphere Transcoding Publisher transforms one form of content into another so that it can be presented on a device that is different from the originally intended target. IBM WebSphere Transcoding Publisher performs this transformation automatically and on-the-fly, reducing or eliminating the need to maintain multiple versions of the content. A good example of this is changing HTML content that is intended for desktop PCs into WML content that is suitable for displaying on a WAP-enabled mobile phone.

IBM MQSeries Everyplace enables pervasive devices to participate in commercial messaging, sending messages between applications, and assuring their delivery (once and only once), in a secure and highly efficient manner, operating in both connected and disconnected modes.

IBM Everyplace Synchronization Manager is another complementary technology that allows pervasive devices to work in semi-connected modes. It enables pervasive devices to operate applications offline, and to synchronize the results of their activities with a server database when connectivity is re-established.

Optimization

As with all e-business solutions, performance is a critical issue. If the performance is poor, it is easy for a customer to go elsewhere. Customer expectations are constantly rising, and as the your solution increases in popularity, it must be capable of scaling to cope with the increased demand. This is provided through the inclusion of caching and load balancing support to ensure high performance and scalability.

Subscriber and device management

In the world of pervasive computing, a single device may have multiple users, and a single user may access the network using multiple devices. IBM WebSphere Everyplace Suite includes Tivoli technology to manage subscribers and their devices, easing the burden of administration and system maintenance.

Tivoli Personalized Services Manager (TPSM) provides a comprehensive set of management services to the solution, including content personalization, subscriber enrollment, customer care and self-care, report generation, and interfaces to external billing systems.

Supporting components

A number of supporting components that come with the Everyplace Suite may need to be installed:

- Secureway Directory - Lightweight Directory Access Protocol (LDAP) server
- IBM HTTP Server
- IBM DB2 Universal Database Enterprise Edition
- WebSphere Application Server Standard Edition
- Java Development Kit (AIX only)

5.1.2 Considerations for m-commerce

Depending on the environment, there are several different ways in which WebSphere Everyplace Suite can be deployed to extend a WebSphere Commerce Suite store for mobile devices. Not all of these involve direct integration between the Everyplace Suite and the Commerce Suite. Described here are three example deployment scenarios.

Internet Service Provider and merchant - scenario 1

A typical customer for the WebSphere Everyplace Suite is an Internet Service Provider (ISP). In this case, Everyplace Suite is most likely being used to form the core of the ISP's infrastructure, covering device connectivity right through to subscriber management. As such, it is likely that every component within the Everyplace Suite is deployed. The ISP is providing its subscribers with the means to access any content provider's content from whatever device they choose to use.

In this scenario, the content provider (commerce Web site), is dependent on the ISP to provide the wireless middleware technologies to support mobile clients. The only consideration here for the merchant is that the Commerce Suite application, or store, must be able to support requests from these other types of devices. This involves making use of the pervasive extensions in WebSphere Commerce Suite V5.1.

This scenario will work for devices using standard content retrieval methods, such as requesting pages of content from the Internet using a HTTP request. For devices using a more specialized method, such as asynchronous messaging or database synchronization, the merchant and ISP would have to collaborate to ensure that the appropriate access permissions have been given.

Internet Service Provider and merchant - scenario 2

This scenario assumes a closer relationship between the merchant and ISP. For example, a merchant may collaborate with a major ISP to make their store available exclusively to subscribers. This is analogous to NTT DoCoMo working with content providers in Japan to provide i-mode services.

The ISP in this case already has an Everyplace Suite infrastructure in place and a large number of subscribers. Both of these are attractive to the merchant for the following reasons:

- The ISP bears the overhead of the infrastructure maintenance and has the experience to effectively manage the connectivity.
- The large subscriber base also offers an instant, captive market for the merchant to address.

In this scenario there are a number of key challenges to be addressed when integrating WebSphere Commerce Suite into the environment. These are described below.

Subscriber management

Within the Everyplace Suite, subscriber details are stored in LDAP and maintained by the Tivoli Personalized Services Manager (TPSM) product. New users enroll using the application provided by TPSM and the data is then propagated to LDAP.

WebSphere Commerce Suite by default uses its own database to store user information. It is also possible, however, to configure Commerce Suite to use an LDAP source instead (see *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167 for details on configuring WCS to use LDAP). This means that WCS could use Secureway Directory as its LDAP source, and therefore pick up all the pre-stored subscriber information.

Security

Once a client is authenticated within the Everyplace domain, the merchant may want his clients to be able to browse the store as if they had logged in to WCS in the normal way. To enable this to happen, Commerce Suite would have to be able to understand the PvC HTTP headers added to the request by the Everyplace Authentication server.

Within the Everyplace Suite domain, all users are either authenticated by the Authentication Server or the Wireless Gateway. The user credentials are then added into the HTTP request headers so that other applications within the Everyplace Suite domain have this information readily available.

Clients using asynchronous messaging

It is possible that some customers may be using mobile devices, such as PDAs, which have messaging capabilities. In this case, it is possible to deploy MQSeries Everyplace in conjunction with WCS V5.1 in order to support these clients.

Clients using MQSeries Everyplace (MQe) can send messages to Commerce Suite using the MQSeries bridge. It is unlikely that the average shopper would use this approach to access a store, but it does present some interesting possibilities for business-to-business scenarios. For example, reordering of goods could be handled entirely through messages sent to Commerce Suite.

Clients using data synchronization

Some clients may be using devices with greater storage capacity and thus are able to support some sort of database application. The Everyplace Suite includes a product called the Everyplace Synchronization Manager, which provides support for the synchronization of any Open Database Connectivity (ODBC) database on a pervasive device that has a database installed (for example DB2 Everyplace). Clients can therefore synchronize a subset of Commerce Suite's back-end database, such as the product catalog, onto their device for the purposes of offline browsing.

Merchant - scenario 3

In this scenario, we are assuming that the merchant is going to implement his/her own infrastructure to support mobile devices using WebSphere Everyplace Suite. Taking WebSphere Commerce Suite as the starting point, the merchant needs to do the following to make his/her store accessible from mobile devices:

- Produce appropriate device content
- Provide connectivity for mobile devices

The first task is accomplished by producing the relevant JSP templates for the store, then developing and registering the appropriate pervasive adapter(s) within Commerce Suite.

The second task relates to the selection of appropriate WebSphere Everyplace Suite components. Since the merchant already has Commerce Suite providing user management functions, it may be that Tivoli Personalized Services Manager is not required. Login and authentication are also provided by Commerce Suite; therefore, the Authentication Server does not need to be deployed either.

It is likely that a situation will arise where only a select few components of the Everyplace Suite have been deployed. In such a case, we are not implementing a WES authentication method. Instead, we are adding in the connectivity for the different types of mobile devices, and providing content adaptation tools, such as WebSphere Transcoding Publisher.

Note: WebSphere Everyplace Suite V1.1.2 comes with WebSphere Transcoding Publisher V1.1.2. This version of the Transcoding Publisher does not contain the functionality that was added to V3.5. The next version of the Everyplace Suite will include an updated version of the Transcoding Publisher. If you decide to use the stand-alone WebSphere Transcoding Publisher V3.5, bear in mind that this does not use LDAP to store its configuration.

This scenario is more of an exercise in integrating individual products from the Suite rather than a complete integration of the two suites. It is likely that this will be the most common approach for combining the Everyplace Suite and the Commerce Suite.

5.1.3 Where to find more information

- *An Introduction to IBM Everyplace Suite Version 1.1, Accessing Web and Enterprise Applications*, SG24-5995
- WebSphere Everyplace Suite product documentation can be found at:

  `http://www.ibm.com/pvc/tech/library.shtml`

5.2 IBM WebSphere Transcoding Publisher (WTP)

One of the biggest challenges in mobile computing is how to support many different devices, each potentially using their own unique presentation language. Coupled with this, you need to be able to do it in an efficient manner, taking into account low network bandwidths.

5.2.1 WTP overview

IBM WebSphere Transcoding Publisher V3.5 extends the reach of applications by converting content for display on mobile devices. It provides a number of default transcoders for the leading pervasive devices, as well as the capability to write and deploy your own transcoders. Through the use of preference profiles, it can tailor content for lower speed networks by removing images.

Provided here is a brief overview of the transcoding publisher and its components. For more detailed information, please refer to *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965.

Architecture

The WebSphere Transcoding Publisher consists of the following components (see Figure 5-3):

- **A plug-able framework for IBM-provided and custom built transcoder plug-ins**. All plug-ins have access to a standard set of core services, such as the ability to obtain preference profile details.
- **A basic set of transcoder plug-ins**, such as the one that applies Extensible Stylesheet Language (XSL) stylesheets to Extensible Markup Language (XML) documents in order to transform the content.
- **A developer's toolkit**, enabling developers to write custom transcoders. These could be used to support new content types or perform new content filtering functions, for example.
- **Administration services**, so that administrators can control configuration information and preference profiles.

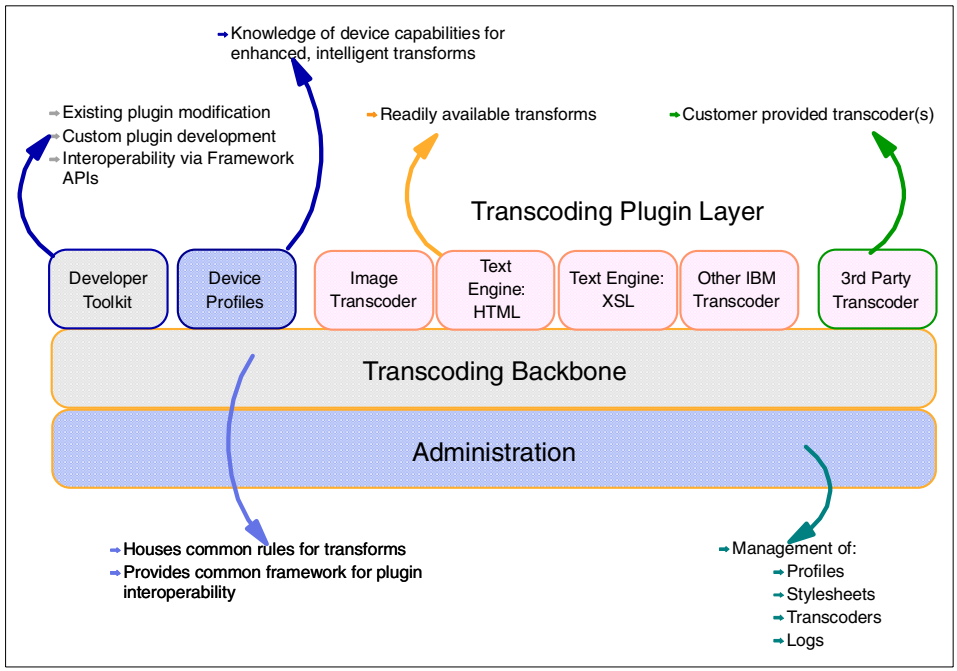


Figure 5-3 WebSphere Transcoding Publisher architecture

Transcoders

A transcoder modifies the contents of a document. IBM WebSphere Transcoding Publisher V3.5 provides a standard set of transcoders (shown in Table 5-1) to convert HTML documents, manipulate images, and transform XML documents to the appropriate format.

Table 5-1 The default enabled transcoders

Transcoder	Function
Text Transcoder	Modifies HTML and XML documents
Image Transcoder	Modifies size and quality of GIF and JPEG images
Fragmentation Transcoder	Fragments documents into pieces that can be displayed on the receiving device
Resource Repository	Stores multiple versions of transcoded documents

The HTML DOM Generator transcoder is disabled by default. This transcoder creates a Document Object Model (DOM) from HTML, enabling content providers to select parts of the HTML for display to the user.

The Annotation Transcoder is also disabled by default. It allows developers to prevent portions of a document from being displayed on a device.

The i-Mode transcoder is not registered when you first install the product. This transcoder allows you to convert HTML documents for display on i-Mode devices.

Preference profiles

IBM WebSphere Transcoding Publisher V3.5 provides a standard set of network and device profiles that enable content customization according to specific device form-factors and network bandwidth constraints. You also have the capability of extending the product capabilities by adding your own profiles.

Device profiles

- Palm Pilot PDA
- Wireless phone - WAP
- Device default
- Wireless phone - HDML models
- Wireless phone - i-mode two-color models
- XML-capable desktop browser
- Windows CE PDA
- Desktop - Netscape
- Wireless phone - i-mode 2 monochrome models
- Wireless phone - i-mode Model 501i
- Desktop - Internet Explorer 4

Network profiles (only used in proxy mode)

- Dial network
- Wireless network
- Network default

Deployment models

IBM WebSphere Transcoding Publisher V3.5 can be deployed in a number of different modes to fulfill different requirements. The modes are:

- Network proxy

- Reverse proxy
- WebSphere MIME filter
- JavaBean

The Server Setup program is run immediately after you install the product, allowing you to select the mode of operation. You can re-run this program at any time to change the mode. The only prerequisite here is that IBM WebSphere Application Server V3.5 be already installed on the same machine in order to use the MIME filter option.

Network proxy

This is the simplest mode, with IBM WebSphere Transcoding Publisher V3.5 acting as an intermediary sitting between the client and the content server. Clients are configured to use the transcoder as a proxy, filtering all the information they receive. Typically, you use the product in this fashion to enhance the content for the client, perhaps adding your company's stock price to the bottom of the page or transcoding the images for display on a mobile phone.

In situations where you don't own or have access to the Web servers that contain the content, you would use the network proxy mode. However, the single biggest disadvantage with this approach concerns security.

The process of transcoding can only take place using data that is not encrypted. This means that you cannot have a secure connection, using SSL for example, between the client and the Web server when using the network proxy. You can overcome this in one of two ways. You can secure the proxy behind the firewall of a trusted party, or you can choose to implement the WebSphere MIME filter mode.

When running as a proxy, you can also specify an HTTP caching proxy, such as IBM WebSphere Traffic Express, for storage of transcoded pages. This is particularly useful when a large number of users are accessing a site using the same type of device, or when large documents are being transcoded.

Reverse proxy

This is very similar in operation to the network proxy, and is used when the client devices cannot be configured to use a proxy. Instead, the transcoding publisher acts as a proxy to Web servers, rather than to clients. Clients access the Web server directly, but are in fact being directed via the Transcoding Publisher. All the considerations for the network proxy also apply to the reverse proxy.

WebSphere MIME filter

The Server Setup program allows you to configure the Transcoding Publisher as a MIME filter for applications running in IBM WebSphere Application Server. When you view your application using the WebSphere administration console, you will see that three transcoding filters have been added to the chosen application. All requests for the application will go through the Transcoding Publisher.

The biggest advantage with this mode concerns security. By transcoding at the source, we are able to employ encryption methods such as SSL because the content is transcoded prior to being encrypted and sent to the client. Another advantage is that the Transcoding Publisher is then managed as part of the WebSphere Application Server, so it controls the starting and stopping of the product.

The disadvantages of this mode are that you can only transcode the content originating from one server and you no longer have access to the Request Viewer tool, which is key to debugging applications.

JavaBean

WebSphere Transcoding Publisher also comes with a set of JavaBean components. These can be included in servlets and JSPs to build custom transcoding applications. This is the most flexible option of the different Transcoding Publisher modes. It does, however, assume that you know the structure of the content that needs to be transcoded. You also need to write your program to pass all the relevant data (HTTP request information, etc.) to the transcoding beans. This functionality is already provided for you in the other Transcoding Publisher modes.

Administration

All the administration for the transcoding publisher is handled through the administration console (see Figure 5-4). You use the console to perform common tasks such as registering new device profiles and transcoders.

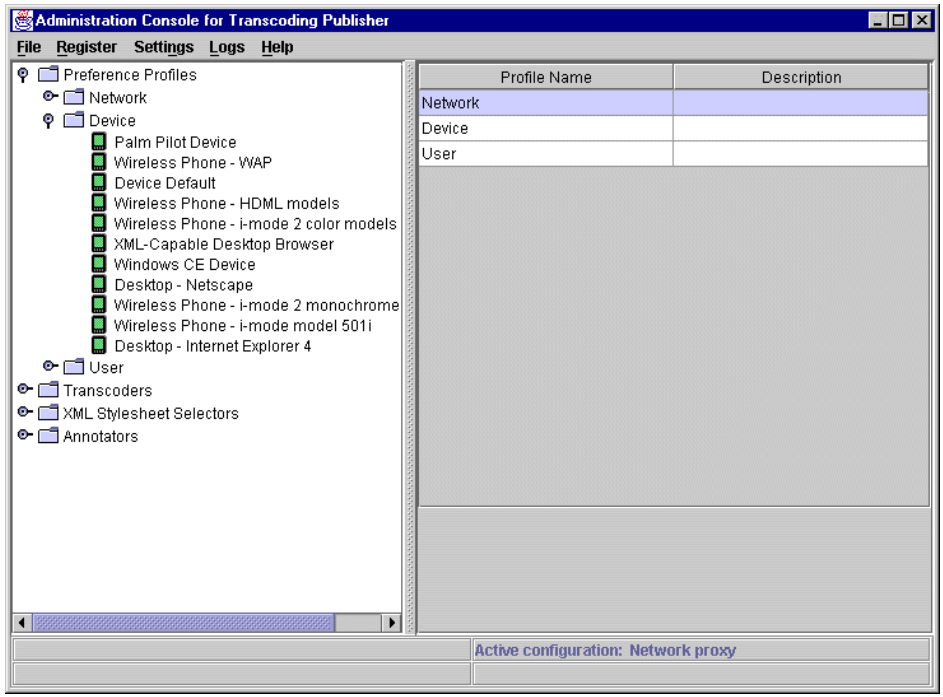


Figure 5-4 WebSphere Transcoding Publisher administration console

Other tasks that can be performed from the console include changing firewall and port settings. You can also specify the amount of log information that is generated by the Transcoding Publisher for debugging purposes.

Problem determination and debugging

WebSphere Transcoding Publisher provides a number of ways to obtain important information that can be used when debugging an application. Full message logging and tracing capabilities can be enabled using the administration console. The most useful tool, however, is the Request Viewer (see Figure 5-5).

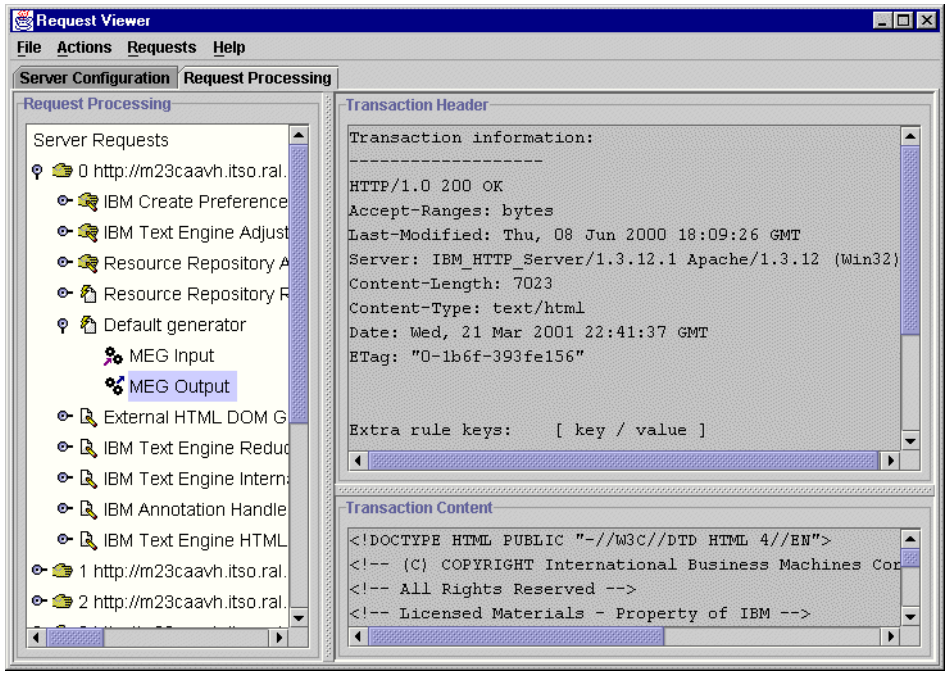


Figure 5-5 WebSphere Transcoding Publisher Request Viewer

This tool is only available when the Transcoding Publisher is configured to run in proxy mode, and you must stop the Transcoding Publisher application in order to run it. This is because the Request Viewer is in fact the Transcoding Publisher running in debug mode, with a graphical user interface.

Using the Request Viewer, developers can view every detail of each incoming request in real time. The transcoders that are invoked for the request are displayed together with their input and output content formats. This tool is an invaluable asset when debugging applications and should always be used to verify that an application is being transcoded in the correct manner.

Note: This tool is available for use only when using the Transcoding Publisher in a proxy configuration (see “WebSphere MIME filter” on page 120).

5.2.2 Considerations for m-commerce

Outlined here are the most common ways in which you could deploy WebSphere Transcoding Publisher in conjunction with WebSphere Commerce Suite.

Proxy

As already outlined in this chapter, running in a proxy configuration means that you cannot support SSL connections between the mobile client and the WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000 server. Since the store by default operates in a secure mode (that is, all pages are requested using the HTTPS protocol), this rules out using the Transcoding Publisher as a proxy for PC clients. There is a way, however, to disable Commerce Suite's internal checking for secure connections from mobile devices. Refer to the *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167 for details. This might be useful in situations where two separate Commerce Suite instances are being used - one to service PC requests and another for mobile devices. In most cases, though, the server is likely to be handling both types of request, as this avoids administering two store IDs.

WebSphere MIME filter

This is the recommended approach when deploying WebSphere Transcoding Publisher V3.5 in conjunction with Commerce Suite. By configuring the product in this fashion, all requests passed to the WCS Stores application can be filtered by the WebSphere Transcoding Publisher. Refer to Chapter 7, "m-commerce runtime environment" on page 141, for detailed instructions on installing and configuring this mode of transcoding.

Attention: After configuring WebSphere Transcoding Publisher as a MIME filter for the WCS Stores application, you may find that you are unable to access the store from a browser. If so, open the WebSphere Commerce Suite Configuration Manager, delete the WCS instance and then recreate it. The application should then be accessible again.

5.2.3 Where to find more information

- *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965
- Product documentation, white papers, articles, etc. can be found at:

  `http://www.ibm.com/software/webservers/transcoding/library.html`

5.3 IBM Everyplace Wireless Gateway (EWG)

This section provides an overview of the IBM Everyplace Wireless Gateway and highlights the considerations for use with m-commerce.

5.3.1 EWG overview

The IBM Everyplace Wireless Gateway (EWG) V1.1.3 is currently bundled with WebSphere Everyplace Suite V1.1.3. Within our test environment, we used EWG packaged with WES V1.1.3 with the EWG service pack that can be downloaded from the following URL:

`http://www.software.ibm.com/dl/everyplace/gateway-p`

The Wireless Gateway extends the corporate network for e-business solutions and protects existing investment in software and information technology infrastructures. It allows secure, optimized access by mobile client devices such as notebooks with wireless modems, personal digital appliances (PDAs), and many WAP-enabled smart phones and communicators over a wide range of wireless and wireline networks, such as local area networks (LANs) and wide area networks (WANs).

Architecture

The Wireless Gateway has two components: the Wireless Gateway itself and the Gatekeeper administration console.

Wireless Gateway

The Wireless Gateway is a communications platform, providing connectivity for wireless networks (both WAP and non-WAP) and wireline networks (such as dial-up). Internally, it is structured in a series of layers that convert the network bearer protocol into a TCP/IP, WAP or WEB stack. The Wireless Gateway also provides enhanced functionality with data encryption, user authentication and optimization, making it more than just a network access router.

Note: Some people use the terms Wireless Gateway and WAP gateway interchangeably - this is incorrect. The Everyplace Wireless Gateway provides connectivity between many different types of networks and supports many different protocols. WAP is one of the main wireless protocols, and the Wireless Gateway can be configured for WAP support. On the other hand, a normal WAP gateway provides support only for the WAP protocol and the underlying mobile bearer networks.

Wireless Gateway Gatekeeper

These are the components of Gatekeeper:
Clients
Wireless Client
WAP Clients

5.3.2 Considerations for m-commerce

When developing an m-commerce store, it is important to test with real hardware before deploying your application. In order to test with real hardware, you will need to configure a wireless gateway for the desired protocol. For example, if you are developing a WAP application, you will need to configure the wireless gateway to provide WAP gateway service. The WAP gateway will allow mobile clients that have connected to the network access to your m-commerce store in the proper protocol.

Using a protocol gateway allows for the testing of session support for devices that do not support cookies. Some gateways provide session support. In the current release of WCS V5.1, all mobile device session control is done via the PvC adapter and WCS database tables.

Note: In order to enable HTTP cookie support in the WAP gateway, you must have defined some users to the gateway. Within a WES environment, these user profiles are stored and maintained by TPSM and the Wireless Gateway looks up the details in LDAP. In the case of a stand-alone gateway, the user profiles are added using the Gatekeeper, in the same way you add any other resource.

Important: The current implementation of the PvC adapter framework within WebSphere Commerce Suite V5.1 means that session support must be handled within the adapter. The only way to avoid doing this is to not use an adapter. This would mean having separate stores for each type of device (for example, one for HTML and another for WAP).

5.3.3 Where to find more information

- For information about fixes for EWG packaged with WebSphere Everyplace Suite, refer to the support Web site at:

  `http://www-3.ibm.com/pvc/products/wes/support.shtml`

5.4 IBM MQSeries Everyplace (MQe)

MQSeries Everyplace (MQe) is a member of the MQSeries family of business messaging products. It is designed to satisfy the messaging needs of lightweight devices, such as sensors, phones, Personal Digital Assistants (PDAs) and laptop computers, as well as supporting mobility and the requirements that arise from the use of fragile communication networks. It maintains the standard MQSeries

quality of service, providing once-only assured delivery, and exchanges messages with other family members. Since many MQSeries Everyplace applications run outside the protection of an Internet firewall, MQe also provides sophisticated security capabilities.

Lightweight devices require the messaging subsystem to be frugal in its use of system resources. Consequently, MQSeries Everyplace offers tailored functions and interfaces appropriate to its customer set. It does not aim to provide exactly the same capabilities as other members of the family. On the other hand, it does include unique functions in order to support its particular classes of users, such as comprehensive security provision, message objects, synchronous and asynchronous messaging, remote queue access, and message push and pull. MQSeries Everyplace is also designed to integrate well with other members of the IBM pervasive computing family - WebSphere Everyplace Server and other family members.

For more information, refer to the MQSeries Everyplace product home page at:

`http://www.ibm.com/software/ts/mqseries/everyplace/`

5.5 IBM DB2 Everyplace

DB2 Everyplace is a small-footprint relational database of about 150 KB. It is designed for low-cost, low-power, small form-factor devices such as personal digital assistants (PDAs), handheld personal computers (HPCs), and embedded devices.

The DB2 Everyplace solution has three components:

- DB2 Everyplace database
- DB2 Everyplace Sync Server
- DB2 Everyplace Personal Application Builder

The DB2 Everyplace database provides a local data store on the mobile or embedded device for storing local data or data from the enterprise. Data stored in the DB2 Everyplace database is queried, updated, and deleted using industry-standard SQL on the mobile or embedded device.

For data synchronization, DB2 Everyplace Sync Server works with the DB2 Everyplace database to synchronize mobile data to and from DB2 Universal Database for UNIX, OS/2 and Windows, DB2 for OS/390, and DB2 for AS/400.

For application development, the DB2 Everyplace developer has a choice of rapid application development tools:

- DB2 Everyplace Personal Application Builder supports building mobile applications for small handheld devices without writing a single line of code.
- VisualAge for Java or VisualAge Micro Edition, featuring the J9 JVM for Palm OS, for Java Development of mobile applications.
- MetroWerks CodeWarrior for C/C++ development.
- Microsoft Visual C++ for C/C++ development.

DB2 Everyplace is available in two editions:

- DB2 Everyplace Personal Edition - includes the DB2 Everyplace database and Personal Application Builder.
- DB2 Everyplace Enterprise Edition - includes the DB2 Everyplace database, Sync Server, and Personal Application Builder.

For more information, refer to the DB2 Everyplace home page at:

`http://www.ibm.com/software/data/db2/everyplace/`

5.6 IBM Mobile Connect

IBM Mobile Connect V2.41 is a pervasive computing technology solution that enables handheld devices such as IBM WorkPads, Palm, EPOC and Windows CE devices to be integrated into Enterprise Solutions.

Mobile Connect allows organizations to directly transfer information from multiple handheld devices directly to corporate systems, without the need to synchronize via the PC. It enables two-way relational database synchronization, two-way file transfer, and the remote installation of applications. IBM Mobile Connect also supports direct synchronization with Lotus Notes and Microsoft Exchange for server-based synchronization of e-mail, calendars, contacts and tasks.

Mobile Connect also supports IBM DB2 Everyplace and Palm PIM applications. DB2 Everyplace includes the DB2 relational database engine and a set of APIs that allow developers to take advantage of DB2 on these platforms. IBM Mobile Connect provides the support for synchronizing relational database information between DB2 Everyplace and a server or mainframe relational database.

All of these transfers are supported over any networking infrastructure that supports IP, for example Wireless GSM, Internet, as well as standard local area networks (LANs) and wide area networks (WANs). Additionally, the product utilizes a VB Scripting engine that enables a company to “plug” its core business logic components directly into their server. Events handled by these components can then be automatically triggered by the captured data that is being transmitted from the remote handheld devices.

By using Certicom's encryption technology (56-bit or 128-bit), IT administrators can trust that data is transferred between corporate systems and mobile devices securely. Mobile Connect also provides user authentication against an internal configuration or through Lotus Domino or Microsoft Exchange servers.

For more information, refer to the Mobile Connect product home page at:

`http://www.ibm.com/pvc/products/mobile_connect/index.shtml/`

6

m-commerce payment solutions

In this chapter, we provide an overview of payment technologies and compare technologies used in a traditional PC browser client and a mobile client. In addition, we discuss how WebSphere Payment Manager V2.2 can be used as part of an m-commerce solution.

The chapter is organized into the following sections:

- Payment technologies overview
- IBM WebSphere Payment Manager

6.1 Payment technologies overview

This section provides an overview of payment technologies.

6.1.1 Payment solutions for PC browser clients

Buying goods over the Internet using a PC has been common for some years now. As a result, there are a number of tried-and-true payment technologies that are in wide use today. In many cases, these technologies have served as a basis for the development of mobile payment solutions.

SET

The Secure Electronic Transaction (SET) specification was developed by Visa and Master Card to define an open standard for the commerce industry. Its primary purpose is to facilitate the secure use of credit card details over the Internet. SET accomplishes this through the use of digital certificates, which are used to identify and validate the cardholder and the merchant during the course of a credit card transaction. It is the task of software vendors to develop applications that support SET, and there are a number of specification books which describe the protocols and programming aspects.

SSL

The Secure Sockets Layer (SSL) protocol, developed by Netscape, is the most widely used method for securing transactions on the Internet. The protocol employs encryption to safeguard information and preserve its integrity, and uses digital certificates for the purpose of identifying each party involved in a given transaction.

Smart cards

Smart cards are another technology in widespread use, where a card is embedded with a microchip. Most credit cards are now produced with these chips, as are telephone payment cards. Smart cards represent the next step in technology after the old magnetic strips, with features such as:

- Increased storage capacity
- Password protection of stored data
- Ability to incorporate a microprocessor to perform encryption

These features complement the SSL and SET technologies, which require secure storage of private keys. In fact, it is possible for a smart card to handle all of the private key management functions, such as storing, generating and using the private keys.

6.1.2 Payment solutions for mobile devices

There are a number of different ways in which payment transactions are made using mobile devices, ranging greatly in terms of complexity and sophistication. It should be noted that this is an area still very much under development and investigation, with a number of different technologies and solutions being field tested at this time. Included in this section are descriptions of some of the most common approaches. These give a good idea of the challenges involved within the mobile environment.

Wireless Transport Layer Security (WTLS)

This is the simplest and most common way of securing a transaction that is made from a WAP device, by encrypting the data. WTLS is functionally equivalent to SSL in the wired network Internet world, and applies encryption to the connection. Unlike SSL, however, which has several different denominations in terms of strength of encryption and capability, most current implementations of WTLS, in both devices and WAP gateways, do not support the use of digital certificates. This is because the Public Key Infrastructure (PKI) support, upon which WTLS relies for certificates, is a new feature in WTLS 1.2, part of the WAP 1.2 specification.

WTLS is only involved in securing the connection between the mobile device and the WAP gateway. It is not responsible for the connection between the WAP gateway and the content server. Instead, this connection is protected by SSL. This means that the gateway is having to manage the protocol conversion between WTLS and SSL, which has security implications. The conversion process means that for a certain amount of time, information is not encrypted, prior to being re-encrypted for the SSL connection. This has led to a reluctance by many businesses, financial institutions in particular, to provide secure services over WAP.

In cases where the gateway is private, that is managed by the business itself, this is not a problem. In many cases, however, the WAP gateway is public, hosted by some third-party company on the Internet. The mobile phones have been provisioned to use that gateway only. This constitutes a potential risk that is most often seen as unacceptable. This problem is overcome in the next implementation of the WAP specification, which allows the use of redirects, so that a request made to a public WAP gateway can be redirected to a private gateway under the control of the bank or merchant, for example.

Dual-Slot phones

Various companies have considered the use of dual-slot mobile phones, that is, phones that have a card reader built into them for your credit card. Nokia and Visa worked together to produce such a handset, and Telecom Italia introduced a mobile banking solution using this device in Italy.

How it works

Figure 6-1 illustrates a transaction being made from a dual-slot mobile phone.

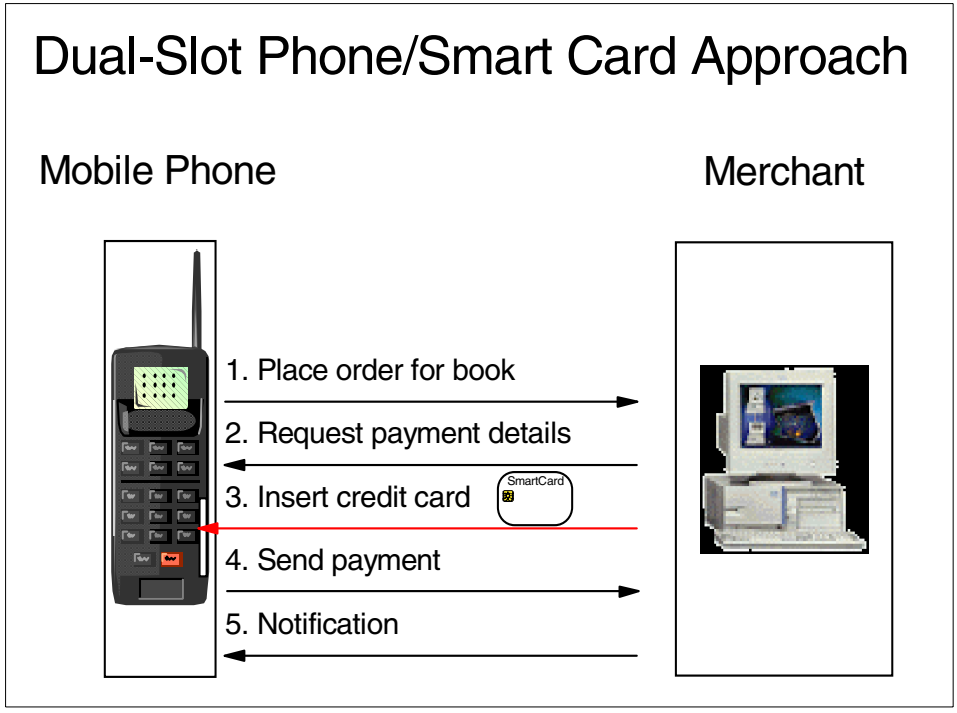


Figure 6-1 Payment transaction using a dual-slot mobile phone

The user accesses the store and submits an order (let's say, for a book). This request is received by the merchant server, which responds with a request for payment details. At this point, the phone prompts the user to insert his/her credit card into the slot in the side of the phone. The card reader interrogates the card and presents the user with the choice of payment method, billing options, etc. The user selects the appropriate settings and submits the information, causing the smart card to be updated with the new transaction. The payment is then sent to the merchant in order to fulfill the order. Finally, a confirmation message is returned to the client to prove that the order was received.

Obviously, this solution relies on specialist hardware, which is fine if you are a business seeking to provide phones to its customers, or if you are a user in the market for the latest type of phone. There are, however, millions of mobile phones already on the market, and such a solution would not be applicable to any of these devices.

Mobile wallet

Rather than having the information held directly by the user for input into the device, another approach is to host this information on a secure server somewhere. This is often referred to as having a wallet on a remote server, and is common practice in many of the original Internet payment solutions.

A typical mobile scenario is shown in Figure 6-2.

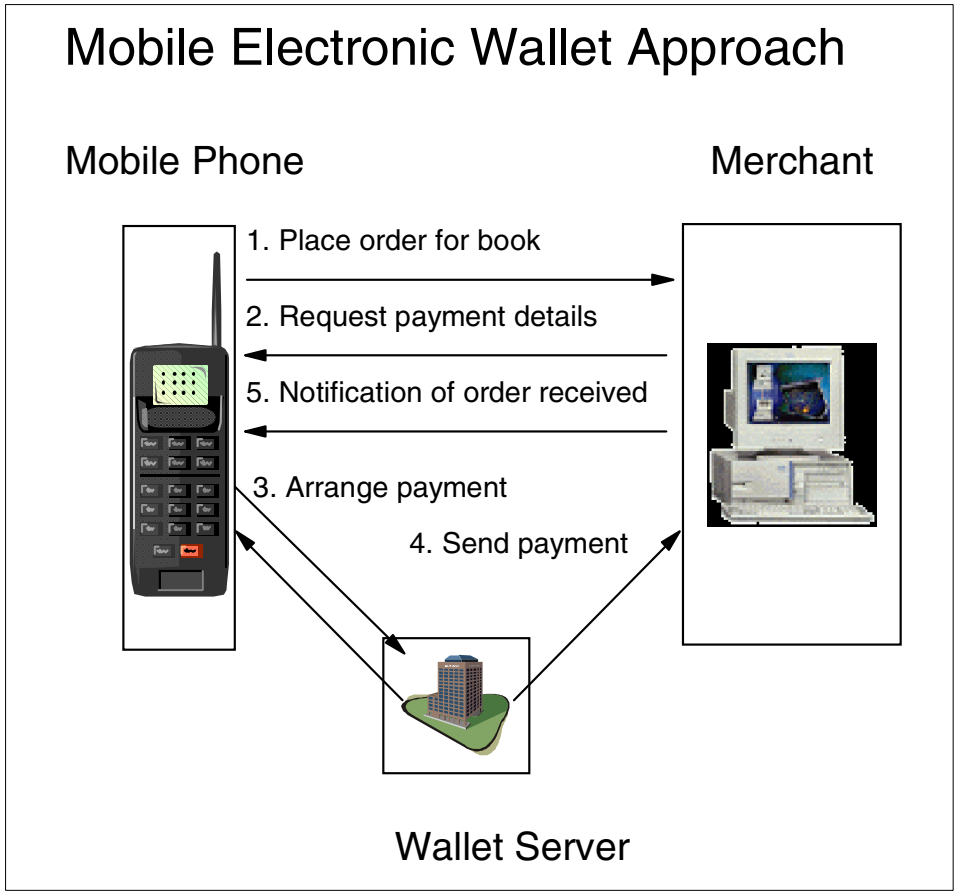


Figure 6-2 m-commerce wallet approach

If a user wants to purchase a product (for example, some cinema tickets), he/she goes to the merchant's store and submit an order. In response, the merchant's server sends back a request for payment information. The phone, upon receiving this request, initiates a dialog with the user's wallet server. The user is prompted to enter his/her password, after which he/she can then specify the payment method and even the address that the ordered goods are to be delivered to. Once the user has completed filling in this information, the wallet server separately contacts the merchant server and sends it all of the payment details. The merchant server then finally returns a notification to the user's mobile phone to acknowledge the purchase.

This is potentially one of the most secure methods of payment, where none of the user's details are being broadcast directly across the Internet. The implication here is, of course, that the merchant supports the use of the wallet server, and has some sort of private line connection to it. This method is not commonly used in the PC environment because of the general lack of knowledge about methods of payment other than sending your credit card details over the Internet. Such a system is ideally suited for the mobile environment, but mobile-phone users are also some of the least computer savvy, and it will require some education in order for such technologies to become widely adopted.

SIM-based application

A third possibility for payment using a mobile device is the use of a GSM mobile phone user's Subscriber Identity Module (SIM) card. Obviously, this solution is specific to this type of mobile phone technology. However, with millions of GSM phones being sold every year and with GSM technology due for imminent launch in the USA, this could be the solution that ultimately prevails.

Figure 6-3 illustrates how the solution works using the SIM-based approach.

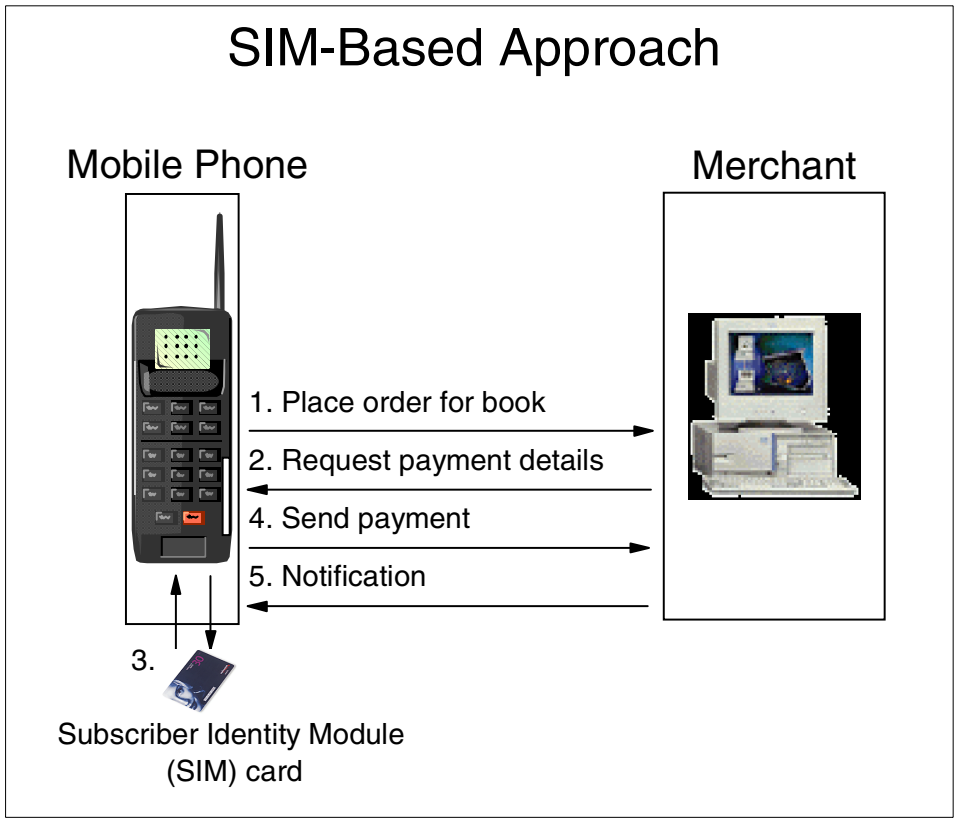


Figure 6-3 SIM-based approach

The initial communication between the phone and the merchant server is the same for all the approaches. The user decides to buy something and submits an order to the merchant server, which responds with a payment request. In this case, however, this request triggers an application on the user's SIM card, which holds the user's credit card details. The user is prompted for a personal identification number (PIN) for security and they can then specify all the necessary information for this purchase. This information is then sent to the merchant server and the order is completed. Again, the merchant can send back a notification to the phone to acknowledge receipt of the payment.

This approach could prove to be one of the most popular payment methods using a mobile device, or a mobile phone in particular. However, there is some work to be done to set this up. Service providers and credit card companies would have to work together to produce the SIM cards with these applications. This could be provided to new users, but in the case of existing mobile phone owners, there would have to be some process that could be followed in order for the users to get their SIM cards updated to include this application.

6.2 IBM WebSphere Payment Manager

In the beginning, the Internet was designed to provide an open network between scientists. Today, it has evolved into an important medium for online commerce. While electronic commerce is quickly emerging as a leading business model, transacting business on the Internet presents two major security challenges:

- Buyers are wary of launching their vital payment card data into the unknown.
- Merchants have been unable to easily confirm the identity of buyers using payment cards.

6.2.1 WebSphere Payment Manager overview

IBM WebSphere Payment Manager bridges the security issues and provides secure, electronic payment processing to Internet merchants. Based on open, standards-based technology, IBM WebSphere Payment Manager works with payment cassettes to support multiple payment protocols. These protocols include SET, Merchant Initiated SET (MIS), CyberCash, and other third-party payment cassettes. Simply put, Payment Manager interfaces with merchant software systems (for example, order fulfillment and online shopping) and provides cash register-like functionality to manage payment processing. The buyer never interacts with Payment Manager directly, because Payment Manager is behind the Internet merchant's storefront, receiving payments and processing those payments with banks and other financial institutions.

The Payment Manager implements a multi-payment Framework architecture that provides a flexible and extensible way to accommodate payment requirements on the Internet for merchants who need to accept multiple payment methods. The multi-payment Framework separates payment management (the Framework) from specific payment types (the software cassettes) so that each can evolve independently.

The Payment Manager provides a plug-in architecture whereby software cassettes for each payment type attach to the payment Framework. The Framework provides the generic infrastructure functions required for making and receiving payments using any payment type.

Payment cassettes are software applications that conform to the data flow and control conventions of the Payment Manager Framework. Each payment cassette contains the specifications of particular payment methods and protocols.

6.2.2 Supported payment methods

The Payment Manager currently provides a cassette for the SET protocol, CyberCash, and VisaNet, as well as two offline cassettes. The offline cassettes are provided with the framework and enable developers to immediately start building commerce sites that interact with Payment Manager.

Beyond the standard set, cassettes can be written by IBM or by third-party payment system implementors, using the Cassette Developer's Toolkit. IBM supports cassette development and offers detailed instruction to developers interested in writing their own payment cassettes.

Part 2

Setting up your m-commerce environment

7

m-commerce runtime environment

This chapter provides the developer, architect, and specialist information required to install and configure a WCS V5.1 m-commerce runtime environment. A WCS V5.1 m-commerce runtime environment includes all of the system components used for WCS V5.1 runtime for PC browser clients. In addition to the standard WCS V5.1 components, both the m-commerce direct approach and the approach using WTP require the deployment of a PvC adapter. When using the WTP approach to m-commerce, you will need to install and configure WTP V3.5 in the MIME filter mode on the system where WCS V5.1 is installed.

The chapter is organized into the following sections:

- WCS V5.1 runtime environment installation
- WTP V3.5 MIME filter installation
- Configuring WTP V3.5 as a MIME filter
- WCS V5.1 messaging configuration
- Where to find more information

7.1 WCS V5.1 runtime environment installation

This section provides instructions for setting up a development unit test or test runtime environment. If you are not familiar with the systems and runtime architecture of WebSphere Commerce Suite V5.1, or need more information, we recommend that you read the following:

- *Fundamentals, IBM WebSphere Commerce Suite V5.1*, product guide found on the WCS V5.1 CD
- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167

For detailed instructions on installing WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000, refer to the following:

- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167

Note: Make sure that you publish the InFashion sample store, shipped with WCS V5.1, prior to setting up the development environment (VisualAge for Java).

- *Installation Guide, IBM WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000*, product guide found on the WCS V5.1 CD

Note: Using the installation procedure documented in this guide is recommended for users who are not familiar with WCS V5.1, WebSphere Studio, or VisualAge for Java.

7.1.1 ITSO test runtime environment

Within our test runtime environment, we used the following software and hardware.

Software used in our test environment

- Microsoft Windows operating systems:
  - Windows NT V4.0 Server + Service Pack 6a - 128-bit encryption
  - Windows 2000 Server + Service Pack 1 - 128-bit encryption

  Note: We developed the procedures and tested on both Windows NT and Windows 2000. We did not find differences in the installation process between Windows NT and Windows 2000. In the following installation descriptions, references to “Windows” can be taken to refer to either version.
- IBM HTTP Server V1.3.12

- IBM DB2 Universal Database V7.1, Enterprise Edition for Windows + DB2 V7.1 Fixpack 2a
- IBM WebSphere Application Server V3.5, Advanced Edition + WAS V3.5 Fixpack 2 + WAS V3.5.2 E-fixes for WCS V5.1
- IBM WebSphere Commerce Suite V5.1, Pro Edition for Windows NT and Windows 2000
- Microsoft Internet Explorer V5.5 + Service pack 1
- WebSphere Transcoding Publisher V3.5

Note: Other Web servers and database software may be used, as documented in the *Installation Guide, IBM WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000*, shipped with the product.

Hardware used in our test environment

This section describes the hardware used within our test WCS runtime environment for Windows NT and Windows 2000 1-tier configuration.

Note: For a 1-tier installation, we recommend that new users refer to the *Installation Guide, IBM WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000*, shipped with the product.

- IBM xSeries Server 230 (8658)
  - 1 GHZ PIII CPU
  - 1GB RAM
  - 18 GB Hard disk
  - 1 Ethernet (built-in)
- IBM PC 300PL (6565)
  - 733 GHZ PIII CPU
  - 512 MB RAM
  - 20 GB DASD
  - 1 IBM Etherjet 100/10
- IBM Netfinity 7000 M10 (8680)
  - 450 MHZ XEON 4-way CPU
  - 2 GB RAM
  - 36 GB DASD
  - 1 IBM Etherjet 100/10

7.1.2 WCS V5.1 high-level installation steps

This section describes the high-level installations steps required to install and configure an m-commerce runtime environment. We will install and configure a 1-tier runtime environment, for the purpose of m-commerce application development and unit testing.

Note: We do not recommend a 1-tier configuration for a production runtime environment. Detailed instructions for 2/3-tier and multi-node configurations can be found in the *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167, and the *Install Guide for WebSphere Commerce Suite V5.1,* shipped with the product.

To install and configure a WCS V5.1 1-tier m-commerce runtime environment, complete the following steps.

Step 1: operating system

1. Install one of the following operating systems within your test runtime environment:
   - Windows NT 4.0 Server + SP6a
   - Windows 2000 Server + SP1
2. Log on as admin user to local User Manager (not Windows domain)

Step 2: HTTP Server

3. Install IBM HTTP Server V1.3.12
4. Configure HTTP admin user
5. Configure SSL
6. Verify configuration for HTTP and HTTPS

Step 3: DB2 Server

7. Install DB2 V7.1 Server
8. Install DB2 V7.1 Fixpack 2a
9. Update JDBC level - usejdbc2
10. Verify configuration

Step 4: WebSphere Application Server

11. Install WebSphere Application Server V3.5 AE (WAS)
12. Install WAS V3.5 Fixpack 2 (V3.5.2)
13. Apply WAS V3.5.2 E-fixes required by WCS

- Copy e-fixes to \<WAS_install_path>\lib
- Update \<WAS_install_path>\bin\admin.config classpath with e-fixes

14. Update Java security \<WAS_install_path>\jdk\jre\lib\security\java.security
15. Create WAS repository database
16. Start the IBM WS AdminServer Windows service
17. Start WAS Default Server
18. Add 443 virtual host aliases
19. Verify WAS by using snoop or other test application
20. Stop Default Server to conserve memory

Step 5: WebSphere Commerce Suite

21. Install WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000
22. Restart the WCS server

Step 6: Create a WCS instance

23. Start the WCS Configuration Manager
24. Create a WCS instance
25. Disable caching by deselecting the Cache Enabled checkbox from the Caching Subsystem tree within the Configuration Manager

Note: At the time of writing this redbook, it is a requirement that you disable the WCS cache. If you do not disable the cache, JSPs served from different directories with the same name can not be distinguished by the WCS cache. The first JSP cached with a given name will potentially be loaded for clients of a different type. For example:

- PC JSPs in root directory of store
- WAP WML JSPs in wml_jsp directory off the root

The StoreCategoryDisplay.jsp contains HTML for the PC version and WML for the WAP version. If the WAP client accesses the JSP first, it is compiled and cached. Next the PC client accesses the store and will not get the cached version of the file instead of the correct version for the detected device type.

26. Stop/restart WCS server
27. Verify environment

Step 7: Deploy InFashion sample store

This step is required for the proper installation and configuration of WebSphere Commerce Studio (VisualAge for Java - WTE).

28. Start WCS Store Services
29. Publish the InFashion sample store
30. Verify that the runtime environment is working properly by using the InFashion sample store for testing purposes

Note: The store must be called InFashion. This is a requirement of the WebSphere Commerce Studio, which will be set up in the next chapter.

Step 8: WebSphere Transcoding Publisher (optional)

This step is required only if you are using the WTP approach to m-commerce. Details on this procedure can be found in 7.2, “WTP V3.5 MIME filter installation” on page 146.

31. Install WTP

    For details, refer to 7.2, “WTP V3.5 MIME filter installation” on page 146.

32. Configure WTP as a filter

    For details, refer to 7.3, “Configuring WTP V3.5 as a MIME filter” on page 149.

Step 9: Configure WCS V5.1 messaging (optional)

This step is required only if you are using the messaging services of WCS V5.1 (for example, to send e-mail or for SMS).

33. Configure the WCS V5.1 messaging services

    For details, refer to 7.4.1, “Configure the WCS V5.1 messaging services” on page 155.

34. Define the message content

    For details, refer to 7.4.2, “Define the message content” on page 156.

7.2 WTP V3.5 MIME filter installation

WTP V3.5 must be installed on the same machine as WCS. For information related to the WTP installation steps, please refer to *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965.

To install and configure the IBM WebSphere Transcoding Publisher V3.5 as a MIME filter, complete the following steps:

1. Install WTP V3.5.
2. Stop the Request Viewer, if already running.
3. From the administration console of WTP choose **Settings** -> **Server Setup**.
4. You will be prompted to confirm your choice. Click **Yes** to continue.
5. When the Transcoding Publisher Server Setup window appears, as seen in Figure 7-1, you will need to start the IBM WS AdminServer service if it is not up and running. If it is stopped, start it before proceeding. Then select **WebSphere Application Server filter,** and click **Next** to continue.

Note: Configuring WTP V3.5 as a MIME filter requires that the IBM WS AdminServer service be running.

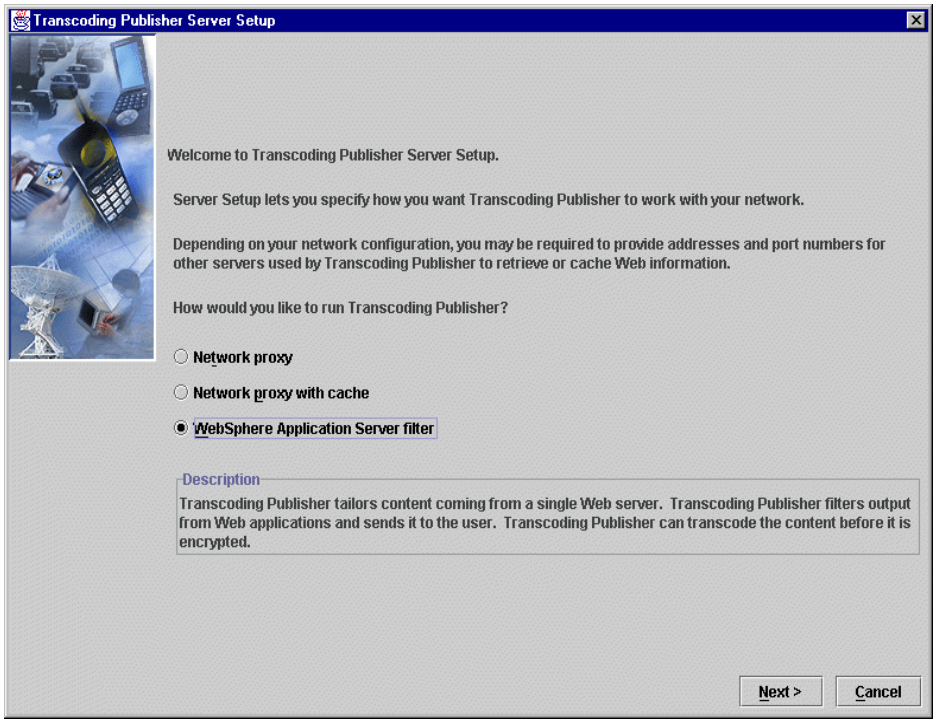


Figure 7-1 WTP 3.5 setup - deployment options

6. When the window seen in Figure 7-2 appears, you will see a list of all available Web applications found in the WebSphere Application Server

repository. In this list, find the two Web applications that are part of IBM WebSphere Commerce Suite V5.1 that you would like to transcode.

– Check **WCS Stores**

  This is invoked when your store is accessed through HTTP(S).

– Do not check WCS Tools

  This is used by the administration tools of WCS V5.1 (for example, the administration console, Commerce Suite Accelerator and Store Services). Do not check, since we only need to transcode the output of the store.

– Click **Next**.

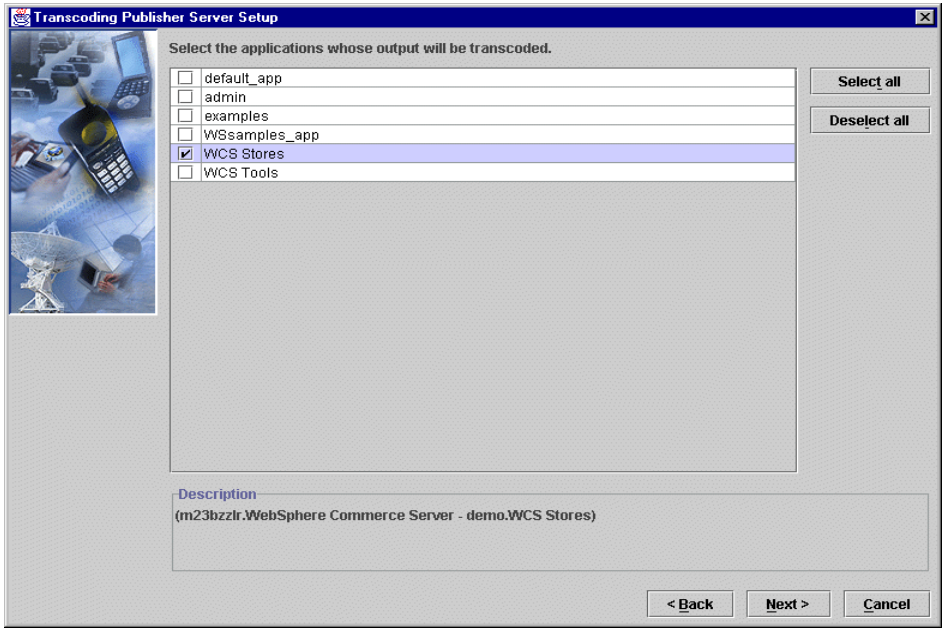


Figure 7-2 WTP V3.5 setup - selecting the Web applications

7. An information window will appear, as shown in Figure 7-3. Click **OK** to continue.

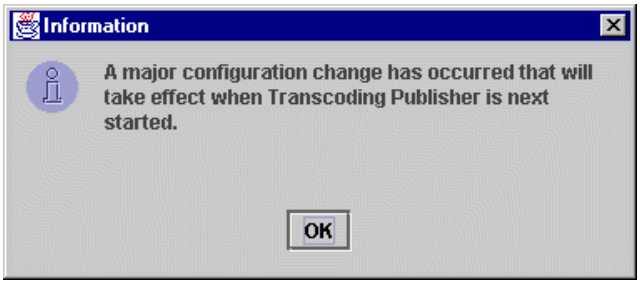


Figure 7-3 WTP 3.5 setup - Information window

8. At this point, you must wait until the configuration process is completed. It will take a few minutes. A message will inform you that the administration console will shut down. Click **OK** to shut down.
9. Once WTP has been configured as a MIME filter, you can access its administration console to customize the way it performs the content adaptation.

The WebSphere Transcoding Publisher MIME filter mode install and configuration are now complete.

Note: We found that occasionally we needed to delete and recreate the WCS instance if transcoding was not working properly.

7.3 Configuring WTP V3.5 as a MIME filter

WTP 3.5 must be installed on the same machine where WCS is also installed. For any information related to the WTP installation steps, please refer to *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965.

In order to configure IBM WebSphere Transcoding Publisher V3.5 as a MIME filter, proceed as follows:

1. Stop the Request Viewer, if already running.
2. From the administration console of WTP choose **Settings->Server Setup**.
3. The system will ask you to confirm your choice. Click **Yes** to continue.

Note: Configuring WTP 3.5 as a MIME filter requires that the IBM WS AdminServer service be running.

In the first window of the server setup, shown in Figure 7-1 on page 147, you have to choose which deployment options you intend to use. Configuring the Transcoding Publisher server as a filter requires that IBM WS AdminServer service be up and running. If it is stopped, start it before proceeding. Then select **WebSphere Application Server filter** and click **Next**.

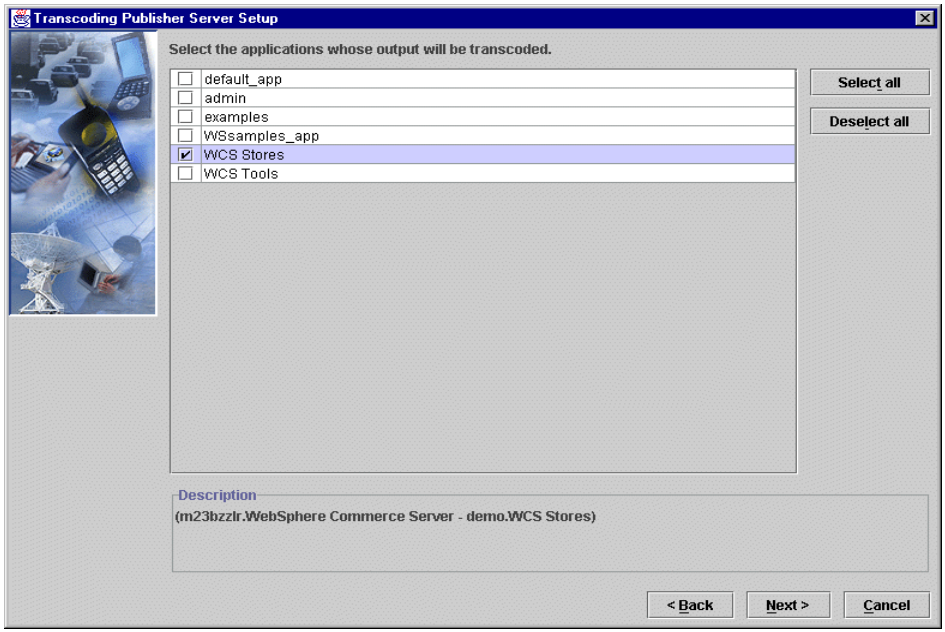


Figure 7-4 WTP V3.5 setup - Select the Web applications

4. In the next window, you can see the list of all available Web applications on your machine. In this list, find the two Web applications that are part of IBM WebSphere Commerce Suite V5.1:
   - WCS Stores
   - WCS Tools

The first one is invoked when your store is accessed through HTTP(S). The second application is used by the administration tools of WCS 5.1 (for example, administration console, Commerce Suite Accelerator and Store Services). Do not check WCS Tools, since you just need to transcode the output of your store. Select **WCS Stores**, click **Next** and then **Finish**. An information window will appear, as shown in Figure 7-3. Click **OK** to continue.

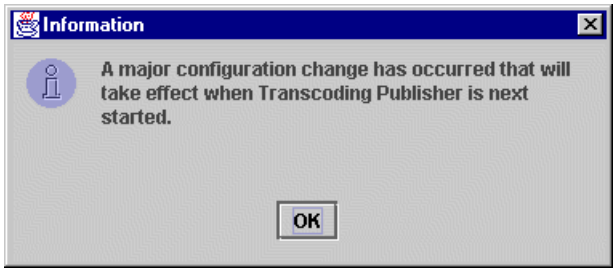


Figure 7-5 WTP 3.5 setup - Information window

5. At this point, you must wait until the configuration process has completed. It will take a few minutes. A message will inform you that the administration console will shut down. Click **OK** to shut down.

Once WTP has been configured as a MIME filter, you can access its administration console to customize the way it performs the content adaptation.

Note: If you are using WTP to transcode XML content, all the required XSLs must be registered through its administration console.

Figure 7-6 shows the administration console. For any information on how to actually customize the transcoding process please refer to *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965.

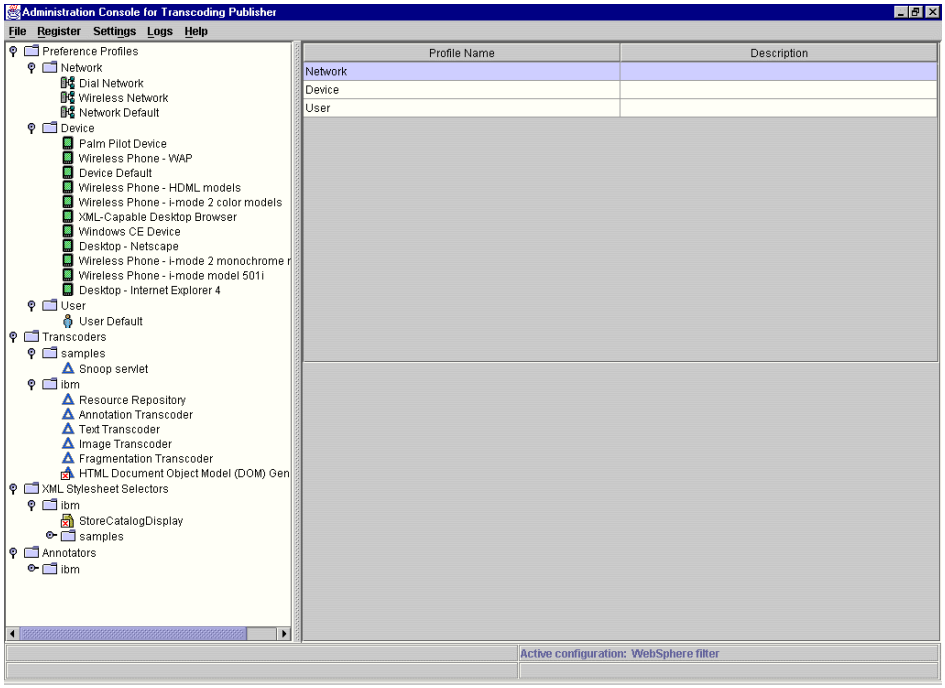


Figure 7-6 Administration console of IBM WebSphere Transcoding Publisher V3.5

Note: There is a problem with the current implementation of MIME handlers in WAS 3.5.2. When a MIME filter is configured, as in the case of our configuration for WTP, HTTP redirects do not work. This means that all the redirects from HTTP to HTTPS performed by the WCS framework for secure commands will not work. One possible workaround is to use HTTPS from the beginning. In fact, in this case no redirect is needed. This problem has been identified but at the time of writing no solution was available.

Tip: With the proposed architecture, all requests coming from a PC browser are also processed by the Transcoding servlet, even if you can configure WTP not to modify anything in the retrieved pages. If you use XML for wireless devices and you do not want WTP to be invoked for requests coming from PC browsers, you can disable the filtering of *text/html*, *image/gif*, *image/jpeg* in the administration console of WAS.

During our testing we have also experienced a problem when using WTP as a MIME filter with JSPs containing the <jsp:include> tag. To better understand this problem, take a look at the following example:

Example 7-1 WTP include tag

```
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<B>Hello from the WebSphere Application Server!</B>
<HR>
<FONT size=-1 face=arial>
<I><BR>This servlet demonstrates the ability to develop a pervasive computing
solution using the
<I><BR>WebSphere Application Server. It can be called from any of the following
types of clients:
<jsp:include page="Included.jsp"/>
</FONT>
</BODY>
</HTML>
```

This simple JSP uses the <jsp:include> tag to include the code generated by a second JSP, called Included.jsp, during the processing of the HTTP request. The source code of this second JSP is shown in the next example.

Example 7-2 WTP include tag (Included.jsp)

```
<TABLE Border=2 WIDTH=30%>
<TR ALIGN=center>
  <TH>Client Type</TH>
  <TH>Data Type Returned</TH>
</TR>
<TR>
  <TD align=center>HTML</TD>
  <TD align=center>HTML</TD>
</TR>
<TR>
  <TD align=center>Speech</TD>
  <TD align=center>VXML</TD>
</TR>
<TR>
  <TD align=center>Wireless</TD>
  <TD align=center>WML</TD>
</TR>
</TABLE>
```

Now, if you have installed WTP as a MIME filter and try to access the URL of the first page from a device that does not use HTML as a markup language, only the HTML generated by the inner JSP is transcoded. The next example shows the code generated for a WAP device.

Example 7-3 WAP sample from WTP

```
<HTML>
<HEAD><TITLE>Hello</TITLE></HEAD>
<BODY>
<B>Hello from the WebSphere Application Server!</B>
<HR>
<FONT size=-1 face=arial>
<I><BR>This servlet demonstrates the ability to develop a pervasive computing
solution using the
<I><BR>WebSphere Application Server. It can be called from any of the following
types of clients:
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml><card id="cmgcfp0"><p><table columns="2" align="LL"><tr><td>Client
Type</td>
<td>Data Type Returned</td>
</tr>
<tr><td>HTML</td>
<td>HTML</td>
</tr>
<tr><td>Speech</td>
<td>VXML</td>
</tr>
<tr><td>Wireless</td>
<td>WML</td>
</tr>
</table>
<do name="te1" type="prev" label="Back"><prev/></do>
</p>
</card>
</wml>

</FONT>
</BODY>
</HTML>
```

The generated code is a mix of HTML and WML, and obviously does not work. This problem has been properly isolated and identified but at the time of writing it was not fixed yet. The obvious workaround for this problem is to avoid using <jsp:include> tags.

Note: This problem does not affect the *<%@ include %>* directive.

7.4 WCS V5.1 messaging configuration

In order to send messages from WCS, the messaging service must be configured.

Note: This section is only required if you intended to send messages from WCS (for example, this is required to send SMS messages).

Two major steps are required to set up the messaging service on the WebSphere Commerce Suite server:

- Configure the WCS V5.1 messaging services
- Define the message content

Note: For a description of the features offered in WCS V5.1 for messaging, refer to, “WCS V5.1 messaging” on page 55.

7.4.1 Configure the WCS V5.1 messaging services

To configure the WCS messaging services, complete the following high-level steps:

1. Start the WCS administration console.
2. Log on with a valid user name and password.
3. Choose the **Site** option and click **OK**.

Note: There are two different levels for setting up messaging: the site level and the store level. In order to enable messaging in a store, the specified messaging has to be enabled under the whole site. Under the store, any enabled site messaging could be disabled.

4. Select the Message Types submenu under the Messaging menu.
5. In the next window, there are existing messages assigned to the site. New message types can be added, deleted, and modified.
6. By adding a new message type, or selecting and changing an existing one, a new window will be displayed called the Message Transport Assignment. This window will have the following options:

- Message Type - the selected or added message type.
- Message Severity x to y - 0 to 0 is recommended for e-mail.
- Transport - the specified transport method for the message. Select **E-mail**.
- Device Format - The specified device format for the message. **Standard Device Format** is the default value, and also recommended here.

7. After selecting the e-mail transport, click **Next**. The following window, called Parameters for e-mail, will be displayed with the following options:
   - Host - the host name of the mail server that receives the e-mail.
   - Protocol - e-mail sending protocol (default is SMTP).
   - Recipient - the name of the recipient.
   - Sender - the sender's name.
   - Subject - the subject of the sent e-mail message.
8. Click **Finish**.
9. You have to set up the same message for the store. After logging on the administration console, select the **Store** option.
10. When the Select Store and Language window appears, select your store and the desired language, and then click **OK**.
11. The next window gives the same options for messaging as the site administration. You can add your transport by clicking **Messaging** -> **Transports**, and configure the messages by clicking **Messaging** -> **Message Types.**

7.4.2 Define the message content

The messaging command invokes the related view which runs the registered JSP in the application server.

Messages, like any other views, are mapped to JSPs. These JSPs contain the content of the message. By customizing the JSPs, you can create your own messages.

The notification message views are stored in the VIEWREG table, seen in Table 7-1.

Table 7-1 VIEWREG table columns and descriptions

Column	Description	Sample content
CLASSNAME	Java class name for the view command.	com.ibm.commerce.messaging.viewcommands.MessagingViewCommandImpl

Column	Description	Sample content
DEVICEFMT_ID	The identifier of the device to which the view will be sent.	-3
HTTPS	Secure HTTP required for this view (1 for secure).	0
INTERFACENAME	Java interface for the view command name.	com.ibm.commerce.messaging.viewcommands.MessagingViewCommand
PROPERTIES	Name value pair used by this view.	docname=PasswordNotify.jsp
STOREENT_ID	Store reference number for this view.	0
VIEWNAME	View name for this entry.	PasswordNotifyView

You can change the content of the message by modifying the associated JSP. It is recommended that you use an existing JSP instead of writing your own, because these message JSPs are not trivial to develop. Unfortunately, they are full of inline Java code (there is no raw content). For example, use PasswordNotify.jsp, which sends a short message with the original password of the user, as a base to start from.

7.5 Where to find more information

- *Fundamentals, IBM WebSphere Commerce Suite V5.1*
- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167
- *Installation Guide, IBM WebSphere Commerce Studio V5.1 Professional Developer's Edition for Windows NT and Windows 2000*
- *Installation Guide, IBM WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000*
- *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965

8

m-commerce development environment

This chapter provides the developer with information on the tools and environment required to develop and test m-commerce stores in WebSphere Commerce Suite V5.1, Pro Edition and WebSphere Commerce Studio V5.1, Professional Developer's Edition for Windows NT and Windows 2000.

The chapter is organized into the following sections:

- Development environment
- WebSphere Transcoding Toolkit
- Test environments and tools
- Where to find more information

8.1 Development environment

To help developers customize and develop new functionality and new stores, IBM has provided WebSphere Commerce Studio. Commerce Studio ships in two packages, Developer Edition and Professional Developer Edition. The basic difference between the two packages is that the Professional Developer Edition has not only all the features of the Developer Edition but can also be used to develop business logic using Enterprise Java Beans. We used the Professional Developer Edition for all our examples.

This section provides instructions for setting up a development environment for development and unit testing within the VisualAge for Java WebSphere Test Environment. If you are not familiar with the programming model of WebSphere Commerce Suite V5.1, or need more information, we recommend that you read the following:

- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167.

  This redbook provides several chapters on the programming model, the development environment, and the customization of a store.

- *WebSphere Commerce Suite V5.1 Customization and Transition Guide*, SG24-6174.

  This redbook provides examples of common customization of stores.

- *Programmer's Guide, IBM WebSphere Commerce Suite V5.1*, the product guide included on the WCS V5.1 CD.

- *Fundamentals, IBM WebSphere Commerce Suite V5.1*, the product guide included on the WCS V5.1 CD.

For detailed instructions for installing WebSphere Commerce Studio V5.1, Professional Developer's Edition for Windows NT and Windows 2000, refer to the following:

- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167

  Make sure that you publish the InFashion sample store prior to setting up the development environment (VisualAge for Java). This is detailed in the installation procedure.

- *Installation Guide, IBM WebSphere Commerce Studio V5.1 Professional Developer's Edition for Windows NT and Windows 2000*

  This procedure is recommended for users that are not familiar with WCS V5.1, WebSphere Studio, or VisualAge for Java.

8.1.1 ITSO development environment

Within our development environment, we used the following software and hardware.

Software used in our test environment

- Software documented in 7.1.1, “ITSO test runtime environment” on page 142
- WebSphere Commerce Studio V5.1, Professional Developer’s Edition for Windows NT and Windows 2000
  - WebSphere Studio V3.5, Advanced Edition
  - VisualAge for Java V3.5, Enterprise Edition
  - WebSphere Commerce Studio V5.1 extensions
- XML Editor
  - IBM XML Tools

  and/or
  - XML Spy

Hardware used in our test environment

This section describes the hardware used within our development environment for Windows NT and Windows 2000. For a 1-tier installation, we recommend that new users refer to the *Installation Guide, IBM WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000*, shipped with the product.

- IBM xSeries Server 230 (8658)
  - 1 GHZ PIII CPU
  - 1GB RAM
  - 18 GB Hard disk
  - 1 Ethernet (built-in)
- IBM PC 300PL (6565)
  - 733 GHZ PIII CPU
  - 512 MB RAM
  - 20 GB DASD
  - 1 IBM Etherjet 100/10
- IBM Netfinity 7000 M10 (8680)
  - 450 MHZ XEON 4-way CPU
  - 2 GB RAM

- 36 GB DASD
- 1 IBM Etherjet 100/10

8.1.2 Development environment high-level installation steps

This section provides the high-level steps required to install and configure an m-commerce development environment. We install the Commerce Studio manually in order to verify each component and provide greater understanding of the configuration, which is needed to develop your own store.

To install and configure a WCS V5.1 m-commerce development environment, complete the following steps:

1. Install the WCS runtime environment on the same system as the development environment.

   Refer to 7.1.2, “WCS V5.1 high-level installation steps” on page 144.

2. Create a WCS instance.

   Refer to 7.1.2, “WCS V5.1 high-level installation steps” on page 144.

3. Deploy the InFashion sample store.

   Refer to 7.1.2, “WCS V5.1 high-level installation steps” on page 144.

   Note: The store must be called InFashion. This is a requirement of the WebSphere Commerce Studio.

For details on the remaining steps, refer to the development environment chapter in *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167.

4. Install WebSphere Studio.
5. Install VisualAge for Java.
6. Install WebSphere Commerce Studio.
7. Install XML editor.
8. Configure WebSphere Studio.
9. Configure VisualAge for Java.

Optionally, consult the tips for developers in the above-mentioned redbook.

8.2 WebSphere Transcoding Toolkit

The WebSphere Transcoding Publisher V3.5 ships with many tools to help you to develop applications and to work with WebSphere Transcoding Publisher. This section provides a brief overview of the Toolkit supplied with the product and selected tools available for download. For details about the Toolkit, see *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965.

The WebSphere Transcoding Toolkit includes the following components:

- Transform Tool
- Request Viewer
- Profile Builder
- Snoop MEGlet

Note: The Request Viewer and the Snoop MEGlet are available only when WTP V3.5 is configured as a proxy.

For this reason, you may consider using a utility that can view HTTP requests for debugging purposes.

8.2.1 Transform Tool

To easily show the effect of transcoding, the Transform Tool provides a split-screen view (see Figure 8-1) showing original content and transcoded content side by side. This partly eliminates the need for acquiring a lot of different devices or device emulators. The tool can be used as a working example of how to use the transcoding JavaBeans in an application. Using the current settings of WTP, the tool will show transcoding of HTML, XML and images.

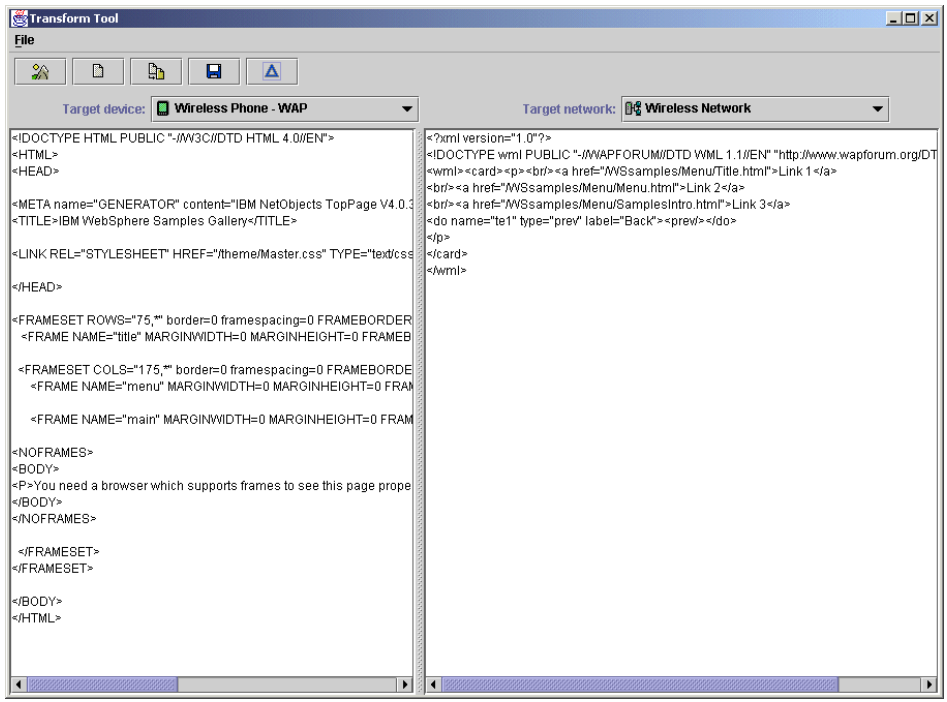


Figure 8-1 Transform Tool - WTP Toolkit

8.2.2 Request Viewer

Request Viewer enables you to monitor the flow of requests through the server, and observe which plug-ins are triggered and when they are triggered (see Figure 8-2). For each transaction, the Request Viewer also displays the header and content information as they are manipulated by the plug-ins. Notice that WTP and the Request Viewer cannot be running at the same time, since the tool will run a complete debugging version of WTP using the same ports.

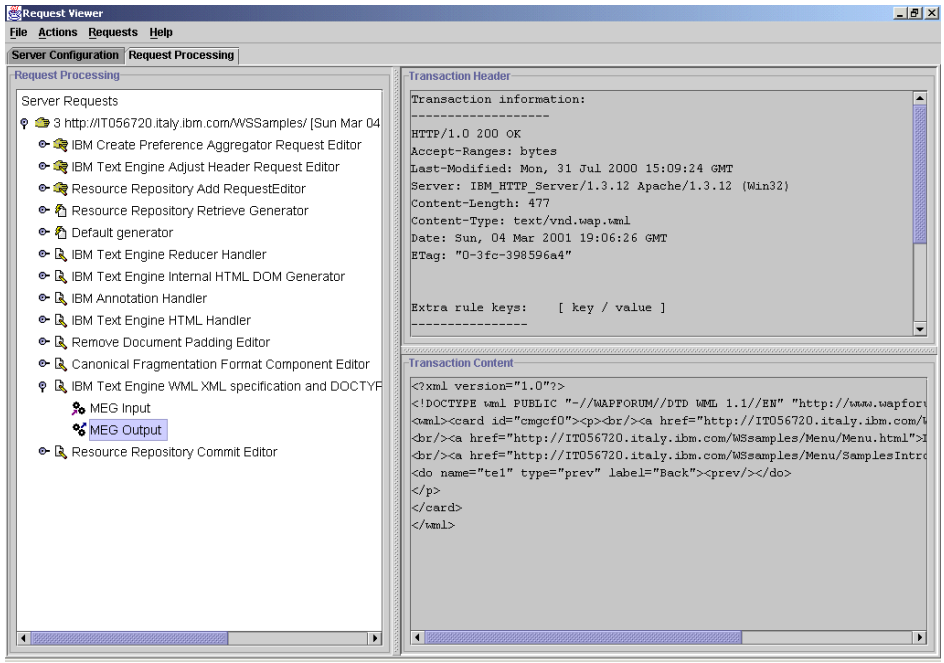


Figure 8-2 Request Viewer

8.2.3 Profile Builder

Profile Builder is a powerful tool to create a new preference profile for a network or a device. The wizard will ask you to:

1. Supply a location to store the new profile
2. Indicate if the profile refers to a device or a network
3. Supply a name and a description
4. Indicate the User-Agent value that can trigger the profile (for example, Windows CE)
5. Choose a set of preferences to include in the profile and, for each of them, indicate if it can be configurable and provide a default value (see Figure 8-3)

Once the new profile has been created, you need to register it from the administration console, by choosing **Register->Preference Profile**.

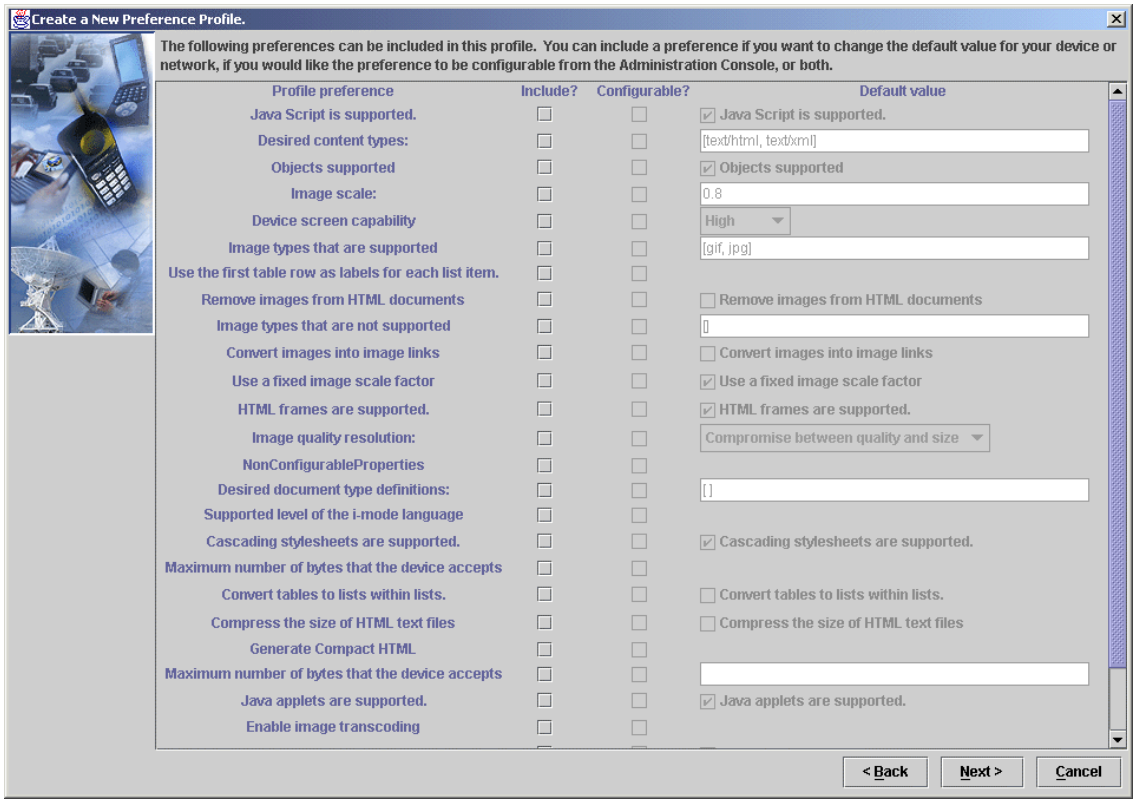


Figure 8-3 Profile Builder

8.2.4 Snoop MEGlet

The Snoop MEGlet is similar to the Snoop Servlet that ships with IBM WebSphere Application Server. It will display all HTTP headers in both the request and the reply. This is a very easy way to spot the User-Agent for any device or browser (the Request Viewer could also be used).

To enable Snoop, proceed as follows:

1. From the WTP Administration Console select **Register -> Transcoder**
2. Point to `<WTP_INSTALL_DIR>\toolkit\meglets\Snoop\Snoop.jar`
3. Supply a name and a description
4. Activate the MEGlet
5. Refresh or restart the Transcoding Publisher Server

Now invoke Snoop by asking for /servlet/Snoop, which is defined as an alias for the servlet in Snoop.prof.

8.3 Test environments and tools

Developing mobile applications presents some unique challenges compared a traditional WebSphere Commerce Suite environment.

We have listed some unique issues for mobile application development:

- How can you test your m-commerce application for wireless devices within your company's intranet?
- How can you debug the m-commerce application for a wireless client?
- How can you develop and m-commerce application when simulators do not provide SSL support required by the m-commerce WCS store?
- What do you need to do to set up an m-commerce test environment?

All the test environments described in this section include WebSphere Commerce Suite V5.1. In addition, if your approach to m-commerce is done via transcoding, you will need to include WebSphere Transcoding Publisher V3.5.

In order to properly test your application before deployment for m-commerce, you need the proper test environment. We have divided the m-commerce test environments into the following:

- Standard PC browsers (Netscape Navigator, Microsoft Internet Explorer)
- Toolkits and simulators
- Real wireless hardware - intranet testing
- Real wireless hardware - Internet testing
- Everyplace Wireless Gateway installation

Each of the environments and tools used for testing provide additional functionality for testing.

8.3.1 Toolkits and simulators

Wireless client simulators provide the application developer with the ability to develop the m-commerce applications for the specific protocol and the targeted features of the device. This allows the developer to take into account such issues as the screen size, controls on the device, and other unique features, such as a color display. The software simulator runs on a PC and allows access to the WCS server device-specific content JSPs, as seen in Figure 8-4.

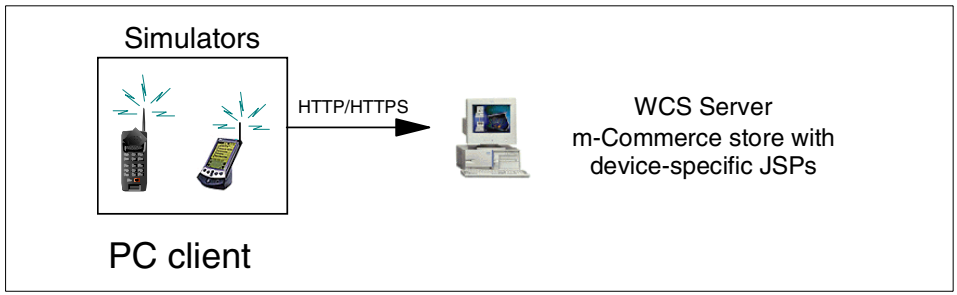


Figure 8-4 Simulator test environment

In addition, we used simulators in test environment going through the IBM Everyplace Wireless Gateway (EWG), which includes a WAP gateway, i-mode gateway, and many other protocols (see Figure 8-5).

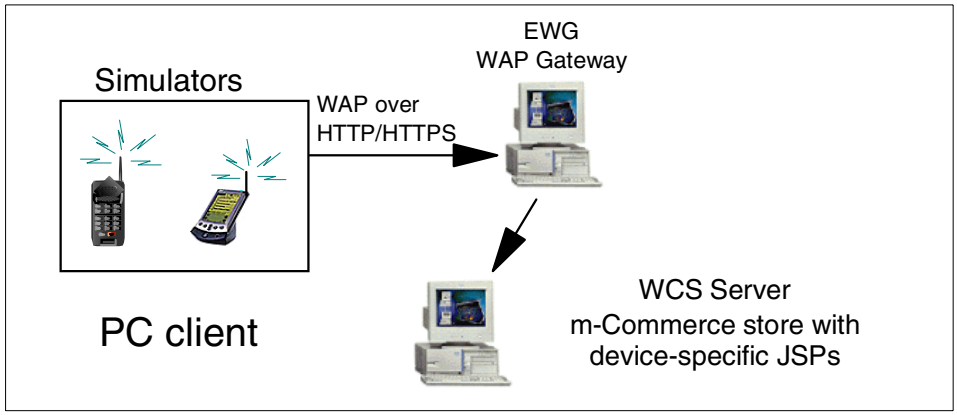


Figure 8-5 Simulator plus EWG WAP gateway test environment

By using the EWG WAP Gateway, we tested our application in an environment more closely related to the customer environment. This test may bring out issues related to session management.

There are pros and cons of using simulators for developing and testing your m-commerce applications. Regardless of the cons, we think that simulators are the logical starting point for developing m-commerce applications.

Pros of using a simulator

- Simulators provide a low-cost method of development.
- Simulators provide a debug console to output code being executed.

Cons of using a simulator

- SSL is not supported by most simulators.

  This becomes a problem when developing m-commerce applications that use SSL. It may be necessary to disable SSL with the WCS database to allow full navigation through your m-commerce store.

- Real hardware is often more current and different from simulators.

  Real hardware testing is required after your initial testing on the simulators to verify the functionality of the specific device with the application.

Simulators

The simulators we used provide unique features, that, used in combination, provide a reasonable level of testing. We used the following simulators:

- UP.SDK V3.2 for HDML

  Refer to 14.1.1, “UP.SDK V3.2 for HDML” on page 266 for more detailed information.

- UP.SDK for WAP V3.2, V4.1

  Refer to 15.1.2, “UP.SDK for WAP” on page 273 for more detailed information.

- Nokia WAP Toolkit V2.1

  Refer to 15.1.1, “Nokia WAP Toolkit V2.1 and simulator” on page 272 for more detailed information.

Tips when using simulators

Here are some useful tips when testing with simulators:

- Turn off SSL in the WCS database

  If your simulator does not support SSL, and you want to test your m-commerce application while developing your application, you will need to turn off SSL in the WCS database.

  Turn off SSL in the VIEWREG and URLREG WCS database tables by executing the following SQL commands:

  a. Start a DB2 command window

  b. Disable SSL in the VIEWREG table as follows:

  ```
  > db2 update viewreg set https=0
  ```

  c. Disable SSL in the URLREG table as follows:

  ```
  > db2 update urlreg set https=0
  ```

- Use a WAP Gateway in your simulator test environment.

By using a WAP Gateway, such as the EWG WAP Gateway, you can resolve problems related to gateways before real hardware testing, such as session management, takes place.

8.3.2 Real wireless hardware - intranet testing

Several variations of testing with real wireless hardware devices can be tried prior to moving to an Internet production environment.

One variation is the use of your company's RAS/NAS together with your own installed and configured Wireless Gateway. In this scenario, your wireless service provider provides the wireless connectivity and device used for testing. The device needs to be able to be configured for your wireless gateway. For example, if you are testing with a WAP device, you need to define your WAP Gateway IP address 9.24.105.101 in the configuration setting of the device, as seen in Figure 8-6.

This test environment requires that you have RAS/NAS gateway configured to access your company's network. For example, within IBM we used the IBM Global Network's (IGN) local access number, and IGN user ID and password to access the IBM Open Net and 9.x.x.x intranet. With the WAP mobile phone configured to use the IBM Everyplace Wireless Gateway (EWG), we were able to access our WCS store. The EWG provides rich support for many wireless protocols and NAS support (for example, WAP). We configured the EWG as a WAP gateway to allow access of the WAP mobile hardware devices for testing purposes, as seen in Figure 8-6.

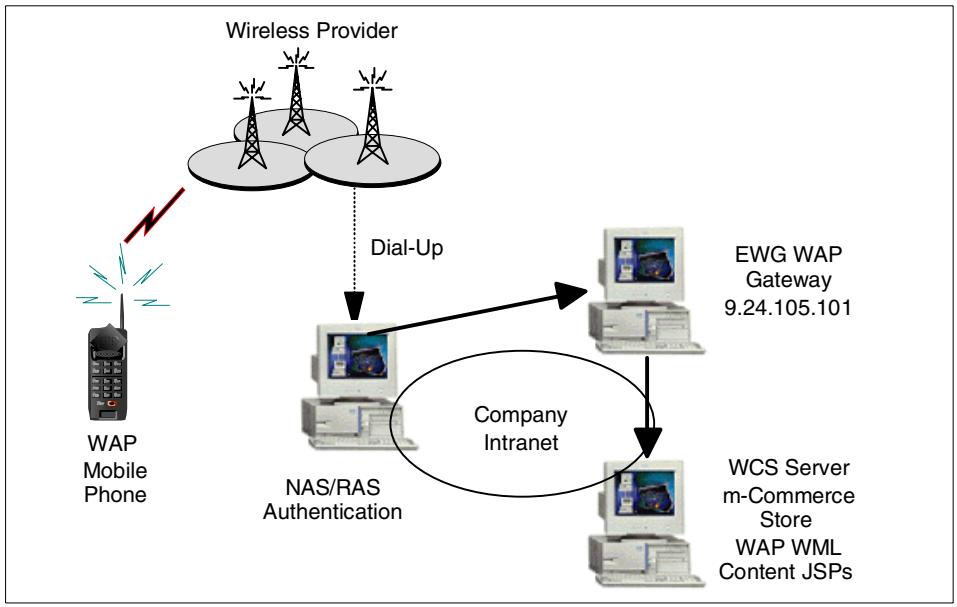


Figure 8-6 Real hardware test on company intranet

If your company does not provide this service, you will need to configure a NAS system such as Windows 2000 Server with a modem attached.

8.3.3 Real wireless hardware - Internet testing

Prior to the production launch of your m-commerce store, you will need to verify that the mobile device hardware and service provider gateway work properly with your store. This can be done in silent launch mode by having a URL that is not provided on your site (manually typed in) with the server accessible on the Internet, as seen in Figure 8-7.

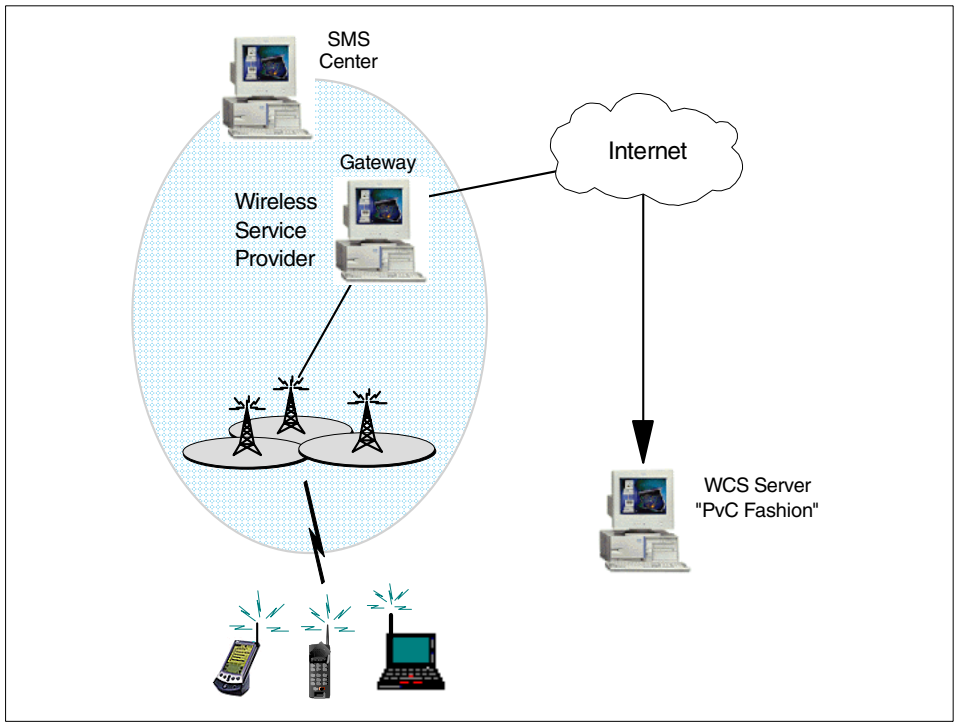


Figure 8-7 Real hardware test environment on the Internet

8.3.4 Everyplace Wireless Gateway installation

IBM Everyplace Wireless Gateway V1.1.3 plus service pack can be run in two modes of operation:

- As a component of WebSphere Everyplace Suite
- As a stand-alone gateway, which requires that you get the EWG service pack

When using the Everyplace Suite installation option, all the implementation details are taken care of. The installation program lets you install the prerequisite products, such as LDAP. You just have to configure the Gateway settings for your implementation.

Stand-alone mode means that the Wireless Gateway is not being used as part of a complete WES solution. In this case there are a number of steps that you need to go through to set up and install a gateway.

Machine configuration

For our environment, we used two RS/6000 systems. We used one system for our LDAP server (host name rs600015) and the other for the Everyplace Wireless Gateway configured for WAP (host name rs60001).

High level EWG install steps

To install the IBM Everyplace Wireless Gateway V1.1.3 provided with WES V1.1.3, plus the EWG service pack, complete the following steps:

1. Download the EWG service pack or contact IBM EWG service for CDs.

   The IBM Everyplace Wireless Gateway (EWG) V1.1.3 is currently bundled with WebSphere Everyplace Suite V1.1.3. We used EWG within our test environment packaged with WES V1.1.3 plus the EWG service pack, which can be downloaded from the following URL:

   `http://www.software.ibm.com/dl/everyplace/gateway-p`

2. Install LDAP on rs600015

   For the purposes of our test setup, we install IBM SecureWay Directory LDAP server direct from the WES 1.1.2 CDs. We use the Everyplace Suite install program to install and configure both LDAP and DB/2. We then verify that LDAP is working properly by running the Directory Management Tool (DMT) and performing a rebind operation. This allows us to view the contents of the directory tree. We can then also stop and restart the LDAP server using the Web administration console, accessible from a Web browser at `http://rs600015/ldap`.

3. Install the Wireless Gateway on rs60001

   The code we install is the latest EWG version (EWG 1.1.3 plus service pack), One of the new features in this updated release is that you can switch the Gateway between modes (WES and stand-alone) just by changing a setting in the configuration.

   The installation of the code is performed using the AIX system management tool called smit. At this point, we cannot use the Gateway until we have installed the Gatekeeper and completed the configuration steps.

4. Install the Wireless Gatekeeper on rs60001

   We also install the Gatekeeper administration console on the same box. Typically you would install this on a separate computer. Again, we use smit to install the Gatekeeper.

Important: Do not run the Gatekeeper immediately after running the installation. Doing so will initiate configuration of the Gateway and the product will assume that LDAP has already been prepared for the installation. In our case it has not, because we need to modify the LDAP schema settings.

5. Configure Wireless Gateway for stand-alone mode

   By default, the Wireless Gateway is always configured to use the WES LDAP schema. When configured to run in stand-alone mode, the Wireless Gateway will apply its own schema to the LDAP server. This schema is a subset of the one provided with the Everyplace Suite.

   To enable stand-alone mode, edit the /usr/lpp/wireless/wgmgrd.conf file. Edit the attribute *wesldaparch*, replacing the default value of 1 (WES schema mode), with 0, as follows:

   ```
   wesldaparch = 0
   ```

6. Configure LDAP

   We now need to modify the LDAP server on rs600015. We installed from the Everyplace Suite install media, meaning that LDAP is configured with the Everyplace Suite schema. The Wireless Gateway will install its own schema, so we need to remove the existing one first.

   Before doing anything else, stop the LDAP server by issuing the following command:

   ```
   # slapd stop
   ```

   Next, drop the LDAP database because this is currently configured for the Everyplace Suite schema, as follows:

   ```
   # ldapucfg -d
   ```

   Having dropped the database, we now recreate it afresh, as follows:

   ```
   # ldapcfg -l /home/ldapdb2
   ```

   The final steps are to create an empty schema file. This is accomplished by deleting the file and then recreating it by editing it using vi and then saving with no contents, as follows:

   ```
   cd /usr/ldap/etc
   rm V3.modifiedschema
   vi V3.modifiedschema
   ```

Important: You must recreate the V3.modifiedschema file after deleting it. This file is used to hold the schema structure and details, and the Everyplace Wireless Gateway will update this file when you run the Gatekeeper for the first time in stand-alone mode.

7. Add an LDAP suffix

   Now that we have recreated the LDAP database and reset the schema file, we need to add a suffix. This will be the root of the directory hierarchy, under which all of our resources and information are stored in LDAP.

   Point a Web browser at the Web admin for the LDAP server using the following URL:

   ```
   http://rs600015/ldap
   ```

   Log in, and select **Settings ->Suffixes** in the left-hand pane. In the Suffix DN field, enter:

   ```
   o=ibm, c=us
   ```

   We have chosen a simple suffix, where `o` is the organization and `c` is the country. You will need this information during the Gatekeeper configuration, so it is a good idea to make a note of the value you specify here.

   Finally, click the **Update** button. The suffix will now appear in the list of Current Server suffixes, as seen in Figure 8-8.

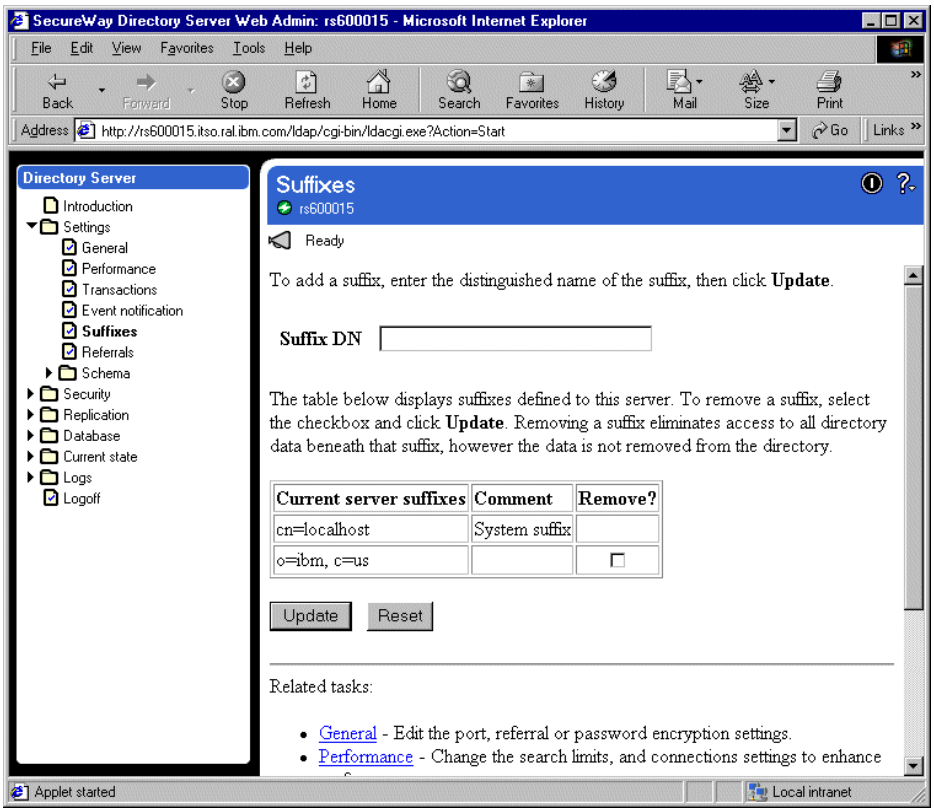


Figure 8-8 SecureWay Directory Web Admin - Suffixes

8. Run Gatekeeper

   With all the preparatory steps completed, we can now run the Gatekeeper program for the first time, using the following command:

   ```
   # wgcfg
   ```

   This will begin the first stage of the Gateway configuration, which is to configure the Access Manager. When this is complete, you will then be able to configure a wireless gateway. Throughout the configuration process, follow all the instructions provided on-screen, making use of the Tips button to provide guidance on what values to enter in the various fields. When prompted for an Organizational Unit, enter your LDAP suffix.

   Next you will need to create a user ID and password for EWG access.

Provided you follow all the instructions, you will now have a fully functional stand-alone Wireless gateway. For additional information on how to install and configure the Wireless Gateway, please refer to the product documentation.

8.4 Where to find more information

- *WebSphere V3.5 Handbook*, SG24-6161
- *WebSphere Commerce Suite V5.1 Customization and Transition Guide*, SG24-6174
- *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965
- *An Introduction to IBM Everyplace Suite Version 1.1, Accessing Web and Enterprise Applications*, SG24-5995
- *Installation Guide, IBM WebSphere Commerce Studio V5.1 Professional Developer's Edition for Windows NT and Windows 2000*
- *Installation Guide, IBM WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000*

9

m-commerce sample store and sample code

This chapter provides information on the m-commerce sample code developed during the project residency of this redbook. Many of these samples are documented in the implementation chapters of this redbook. We provide the code samples and PvC Fashion sample store as a reference of how to develop and deploy an m-commerce store using WCS V5.1.

The chapter is organized into the following sections:

- Download sample store and sample code
- Sample store overview
- Deploy sample store to WCS runtime
- Prepare the development environment

9.1 Download sample store and sample code

This section illustrates a procedure to obtain the sample code and a description of the contents of the .zip file.

Note: The sample code is supplied in the .zip file is used throughout this redbook. The goal of the sample code is to make the development of m-commerce applications easier. We recommend that you download the sample code .zip file and use the contents as a reference.

To download the sample m-commerce code from the IBM ITSO Redbooks Web site, complete the following steps:

1. Download the Web material associated with this redbook, by entering the following in your Web browser:

   `ftp://www.redbooks.ibm.com/redbooks/SG246171`

   Where `sg246171` is the order number for this redbook.

Note:

Alternatively, you can go to the IBM Redbooks Web site at:

`ibm.com/redbooks`

Select the **Additional materials** and open the directory that corresponds to the redbook form number, SG246171.

2. Unzip the SG246171.zip file to the directory of your choice (for example, c:\temp).

Table 9-1 provides a description of the SG246171.zip file contents.

Table 9-1 m-commerce sample code .zip file contents

Directory/filename	Description
\sg246171\PVCFashion\ pvcfashion_en_US_ja_JP.sar	PvC Fashion sample store in WCS store archive format (SAR) * Beneficial for deployment
\sg246171\PVCFashion\PvCFashion.wsr	PvC Fashion sample store in Studio archive format * Beneficial for development

Directory/filename	Description
\sg246171\adapter.pvc\UpPvcAdapter.jar \sg246171\adapter.pvc\UpPvcAdapter.xml	PvC adapter for UP HDML and XML adapter definition of WCS instance XML configuration file
sg246171\adapter.pvc\WapPvcAdapter.jar \sg246171\adapter.pvc\WapPvcAdapter.xml	PvC adapter for WAP WML and XML adapter definition of WCS instance XML configuration file
\sg246171\adapter.pvc\GenericPvcAdapter.jar \sg246171\adapter.pvc\GenericPvcAdapter.xml	PvC adapter for WTP Generic (many device types) and XML adapter definition of WCS instance XML configuration file
\sg246171\adapter.pvc\PrintHttpRequest.jar	JAR containing PrintHttpRequest test program used in creating an adapter
\sg246167\command.pvc	PvC command sample
\sg246167\databean.pvc	PvC data bean
\sg246171\direct\hdml\hdml_jsp \sg246171\direct\hdml\properties \sg246171\direct\hdml\scripts	m-commerce direct HDML sample * JSPs with HDML content * Properties files * Content mgt scripts for HDML
\sg246171\direct\wap\wml_jsp \sg246171\direct\wap\properties \sg246171\direct\wap\scripts	m-commerce direct WAP sample * JSPs with WML content * Properties files * Content mgt scripts for WML
\sg246171\direct\palm\html_jsp \sg246171\direct\palm\webclip	m-commerce direct Palm sample * JSPs with HTML content * JSPs with HTML Web Clipping
\sg246171\wtp\wtp_html	m-commerce using WTP sample * JSPs with simplified HTML content
\sg246171\wtp\wtp_xml	m-commerce using WTP sample * JSPs with XML content

9.2 Sample store overview

While writing this redbook, we used the In Fashion sample store included with WCS V5.1 as a basis to create an m-commerce enabled store called PvC Fashion. The objectives of the PvC Fashion sample store are as follows:

- To provide an m-commerce store that can be deployed

  The deployment process demonstrated will be very similar for other m-commerce stores developed using WCS V5.1
- To provide an m-commerce store as an example for developing the PvC extensions documented in this redbook

9.2.1 PvC Fashion sample store

The implementation of the m-commerce features provided in WCS V5.1 were developed by the IBM Japan Yamato lab. The dominant protocol in Japan is i-mode (cHTML) provided by NTT DoCoMo. The first sample PvC Fashion store was developed for i-mode by the IBM Japan team. Unfortunately, we could not make this code available due to confidentiality agreements between IBM and NTT DoCoMo for i-mode. For information on i-mode and WCS V5.1, contact IBM Japan and NTT DoCoMo.

The next biggest service provider in Japan is KDDI's EZWeb, which provides HDML services. For this reason, we enabled the PvC Fashion sample store for a UP PvC adapter and HDML content.

We then used the HDML content JSPs as a starting point for developing WAP WML content JSPs to browse the catalog. We converted all the HDML content JSPs to WML. However, only the JSPs necessary for browsing the catalog have been debugged and tested. We have included the untested version of the WML JSPs as a head start for developers who may be interested in this code.

In addition, we created Palm HTML content JSPs and used Web Clipping for browsing the catalog. Similarly, we developed simplified HTML and XML content JSPs for WTP.

Features of the PvC Fashion sample store

The sample has the following features:

- Multiple device type support
- Multiple language support (English and Japanese)
- Content-specific JSPs
- Use of the PvC adapter
- Modified shopping flow
- Use of PvC data beans
- Use of PvC commands

The sample shows how to implement a store with support for mobile devices and PC browser clients from one server. This sample supports both English and Japanese languages. It is easy to add other languages by translating the property file, so that multiple languages can be supported on mobile devices.

9.2.2 Web Fashion sample store

A new and more feature-rich version of a sample store, called Web Fashion, has been made available for download from the IBM WebSphere Commerce Suite home page. The Web Fashion store is an enhanced version of the original InFashion sample store, which PvC Fashion is based upon. If you want to start with a more complete sample or template when creating your own store, you may consider using the Web Fashion sample store. The m-commerce sample code provided in the SG246171.zip can be easily modified to work with Web Fashion or other WCS stores.

Refer to the store creation and basic customization sections in the *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167 for detailed instructions on using Web Fashion. The instructions include how to use Web Fashion as a template to create your own store, and how to develop a store using WebSphere Commerce Studio.

9.3 Deploy sample store to WCS runtime

This section includes high-level instructions for deploying the PvC Fashion sample store to your WCS V5.1 runtime. Many of the detailed instructions for these procedures are included in *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167.

Note: We have included instructions for deploying UP PvC adapter and HDML content JSPs, or a WAP PvC adapter and WAP WML content JSPs.

The deployment of both PvC adapters and content simultaneously is supported. However; for the sake of simplicity, the instructions and deployment scripts are written to work for only one at a time.

If you are planning to use more that one PvC adapter at a time, you will need to provide unique values for the deviceFormatId in the adapter definition.

Also, when performing the content management configuration you will need to modify the SQL scripts to provide unique model_id, mdlspec_id, and spec_ids for each adapter.

For more detailed information on deploying multiple PvC adapters, refer to 11.3, “Deploying multiple PvC adapters” on page 222.

We have organized the deployment of the PvC Fashion sample store in the following categories to demonstrate the deployment steps for a standard PC browser-based store, and the additional steps required to deploy an m-commerce store:

- Create a store template
- Create a store from a template
- Publish the PvC Fashion sample store from Store Services
- Deploy PvC adapter and data beans
- Configure content management
- Verify PvC Fashion sample store

9.3.1 Create a store template

The first step in store creation is the generation of a template store from which we will create the actual stores to be customized for m-commerce. For the purpose of our examples, we make a copy of the PvCFashion_en_US_ja_JP.sar store archive to serve as a template.

To create a store template, complete the following steps:

1. Create a directory named PvCFashion in the <wcs_install_path>\samples\stores directory.
2. Copy the PvCFashion_en_US_ja_JP.sar to PvCFashion.sar on the WCS Server.

For example:

From: `c:\temp\sg246171\PvcFashion\PvCFashion_en_US_ja_JP.sar`

To: `c:\ibm\wcs\samples\stores\PvCFashion\PvCFashion.sar`

3. Make copies of the following files in the <wcs_install_path>\xml\sar\ directory:

Original file	Back up to:
catalog.dtd	catalog.dtd.bak
command.dtd	command.dtd.bak
store-all.dtd	store-all.dtd.bak

4. Create a new store owner for PvC Fashion as follows:
   a. Open a DB2 command line processor by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command Line Processor**
   b. Issue the following commands to create a new owner:

```
db2 => connect to <wcs_db> user <dbuser> using <dbuser_password>
db2 => insert into member (member_id,type) values (4444, 'O')
db2 => insert into orgentity (orgentity_id, orgentityname,
orgentitytype) values (4444,'ITSOPVC', 'O')
db2 => commit
```

Note: The `member_id` must be unique. Both `member_id` and `orgentityname` are user defined. The type is the letter O not the number 0.

   c. Close the DB2 Command line processor window.
5. Modify the commit count number of your Commerce Suite instance as follows:
   a. In a text editor or your XML editor, open the file:

```
<wcs_install_path>\instances\<instance_name>\xml\<instance_name>.xml
```

   b. Under the <DevTools tag, set the CommitCount value to 10500.
   c. Save and close the file.
6. When using DB2, increase the log file size of your Commerce Suite database for publishing purposes:
   a. Open the DB2 Control Center by clicking **Start** -> **Programs** -> **IBM DB2** -> **Control Center**.
   b. Expand the **<hostname>** -> **Instances** -> **DB2** -> **Databases** node.
   c. Right-click on your Commerce Suite database.
   d. Select **Configure...**
   e. In the Configure Database window, open the **Logs** tab.

f. In the Logs tab, change the value of the Log file size to at least 2500, and then click **OK**.

g. Exit the Control Center.

h. Stop and Start the WebSphere Commerce Server - <wcs_instance> from the WebSphere Administration Console.

7. Update the Store Services sample list by editing the XML file that identifies the samples, as follows:

   a. Opening the <wcs_install_path>\xml\tools\devtools\SARRegistry.xml file using a text editor or an XML editor.

   b. Add the following entry directly after the <SAR-properties> tag :

```
<SampleSAR fileName="PvCFashion.sar" relativePath="PvCFashion">
   <html locale="en_US" featureFile="PvCFashion/Feature_en_US.html"
sampleSite="RetailModel/preview/en_US/index.html"/>
</SampleSAR>
```

   c. Save and close the file.

You have just created a template from which you can generate other stores.

9.3.2 Create a store from a template

To create a store from the template created in the previous section, complete the following steps:

1. Start Store Services by using Internet Explorer V5.5 (or later) and entering the following URL:

   `http://<your_host_name>/storeservices`

2. In the Store Services Logon page, enter your WCS administrator name and password, then click **Log On**.

3. When the Store Services archive list window appears, click **New**.

4. When the Create Store Archive window appears, enter the following in the appropriate fields and then click **OK**:

   – Store archive: `PvCFashion`

     This is the name of the new store archive. After you have created the store archive, the file extension .sar is added to the name of the store archive, for example, PvCFashion.sar

   – Store directory: `PvCFashion`

     The directory name defines where the Web assets will be published on the server (for example, c:\ibm\wcs\stores\PvCFashion).

– Store owner: From the Store owner pull-down list, select the organization that owns the store.

  For example, we selected the organization `ITSOPVC`, which we created in a previous step.

– Sample: From the sample pull-down list, select **PvCFashion.sar**.

5. You should see a window with the message `PvCFashion.sar created successfully`. Click **OK** to continue.
6. You should now see the PvCFashion.sar listed under the Store archive.

9.3.3 Publish the PvC Fashion sample store from Store Services

Now that a store archive exists for the PvC Fashion store, we need to publish the store from Store Services to the WCS server runtime, so that we can verify our changes as we develop m-commerce code.

1. Click the checkbox next to your newly created SAR file, PvCFashion.sar, and then click **Publish...** from the right-hand side of the window.
2. When the Publish Store Archive window appears, take note of the directory paths specified, accept the defaults and then click **OK**.
3. The Store Archive List should reappear, and the status of the new store should read **Publishing**.

   This process can take 5-10 minutes, depending the specifications of your system.

4. To check the status, click **Refresh** in the right-hand pane.

   Repeat this process until the status says `Publishing completed successfully`.

Note: We recommend viewing the Performance tab in the Windows Task Manager to monitor CPU activity to assist in determining the publishing status.

5. Verify your store from a PC browser client.

   To see the running store, complete the following steps:

   a. Click on the checkbox next to `PvCFashion.sar`, and then click **Publish Summary** in the right-hand pane.

   b. At the bottom of the Publish Summary window, click **Launch the Store**.

Your store will appear in a new browser window.

Tip: It is a good idea to bookmark this page since the URL is very long, and you will want to view your store home page often as you develop.

If you experience problems launching your store, we recommend that you stop/start the WebSphere Commerce Server - <wcs_instance> from the WebSphere Administrators Console. Then repeat the steps to launch the store.

9.3.4 Create an alias for the store

To make it easier for users to access your store, we recommend that you create an alias and an index.html file containing the location.href to your store.

Allow quick access to your store from a Web browser by typing the following URL:

```
http://<hostname>/<alias>
```

For example: `http://m23vnx58/mystore`

Instead of something like the following:

```
http://<hostname>/webapp/wcs/stores/servlet/StoreCatalogDisplay?stor
eId=10001&langId=-1&catalogId=10001
```

Add alias to httpd.conf

To add an alias to the httpd.conf, complete the following steps:

1. Stop the IBM HTTP Server from Windows Services.
2. Change to the directory of the httpd.conf file. For example, c:\ibm\http\conf\.
3. Modify the httpd.conf file. Add the following alias to the path of your published store:

   ```
   Alias /mystore "c:\ibm\wcs\stores\web\"
   ```

Note: After the WCS V5.1 installation, the httpd.conf file can no longer be edited using NotePad (removal of crlf). We used Wordpad to edit the httpd.conf.

4. Save the file.
5. Stop and Start the IBM HTTP Server for the change to take effect.

Create an index.html with a href for your store

To create an index.html file with the location.href for your store, complete the following steps:

1. Change to the directory of your published store. For example, `c:\ibm\wcs\stores\web\`.
2. Create an index.html file in this directory like the one seen in Example 9-1.
3. Change the location.href to match your store settings. For example, change the host name (your storeId and catalogId may be different).

Example 9-1 Sample index.html

```
<html>
<head>
<title>PvCFashion sample store</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script language="JavaScript">
location.href =
"http://m23vnx58.itso.ral.ibm.com/webapp/wcs/stores/servlet/StoreCatalogDisplay
?storeId=10151&langId=-1&catalogId=10151";
</script>
</head>
<body bgcolor="#FFFFFF">
<b><font size="1" face="Arial, Helvetica, sans-serif" color="#333333">InFashion
sample store
<br>One moment.....</font></b>
</body>
</html>
```

Test alias with index.html

Once you have updated the httpd.conf, created the index.html file, and stopped and started the IBM HTTP Server, you can now test the alias. Enter the following URL in a Web browser to access your store:

`http://<hostname>/<alias>`

For example: `http://m23vnx58/mystore`

9.3.5 Deploy PvC adapter and data beans

This section describes the steps necessary to deploy the PvC adapter and data bean JARs and update the WCS instance XML config file with the PvC adapter definition.

Deploy JARs

Complete the following steps to deploy the PVC data beans and PvC adapter:

1. Open a Command Prompt window by clicking **Start** -> **Programs** -> **Command Prompt**.
2. Copy pvcdatabeans.jar to the <wcs_install_path>\lib directory.

   For example:

   From: c:\temp\sg246171\databean.pvc\pvcdatabeans.jar

   To: c:\ibm\wcs\lib

Note: In the future the pvcdatabeans.jar will be included in the WCS V5.1 product. Check for newer versions of this file on the WebSphere Commerce Suite home page at:

`http://www.ibm.com/software/webservers/commerce/wcs_pro/support.html`

The pvcdatabeans.jar includes the PVCDataBufferBean and the UserPVCDeviceBean documented in Appendix C, “PvC data bean reference” on page 355.

3. Copy the PvC adapter JAR to the <wcs_install_path>\lib directory.
   - In UP HDML:

     From: c:\temp\sg246171\adapter.pvc\UpPvcAdapter.jar

     To: c:\ibm\wcs\lib
   - In WAP:

     From: c:\temp\sg246171\adapter.pvc\WapPvcAdapter.jar

     To: c:\ibm\wcs\lib
4. Append PvC adapter JAR and pvcdatabeans.jar to the classpath of the WCS application server.
   a. Start the WebSphere Administration Console by clicking **Start** -> **Program** -> **IBM WebSphere** -> **Applications Server V3.5** -> **Administrator’s Console**.
   b. Select and expand the **WebSphere Administrative Domain** -> **<your_hostname>**.
   c. Select **WebSphere Commerce Suite Server - <wcs_instance>** in the tree view.
   d. Append the following JARs in the command line argument field.

      For example:

      c:\ibm\wcs\lib\UpPvcAdapter.jar;c:\ibm\wcs\lib\pvcdatabeans.jar

      or

c:\ibm\wcs\lib\WapPvcAdapter.jar;c:\ibm\wcs\lib\pvcdatabeans.jar

e. Click **Apply**.

The server will need to be stopped and started for these changes to take effect. We will be making other changes in the next step that will require a stop/restart.

Add PvC adapter definition to the WCS instance XML file

Complete the following steps to insert the PvC adapter definition into the WCS instance XML configuration file with the PVC adapter attribute entries:

1. Stop the WebSphere Commerce Server - <wcs_instance>.
2. Open a Command Prompt window by clicking **Start** -> **Programs** -> **Command Prompt**.
3. Change to the directory of the WCS instance XML configuration file. For example:

   c:\ cd \ibm\wcs\instances\wcs\xml
4. Edit the WCS instance XML configuration file (<wcs_instance_name>.xml). Insert the PvC adapter definition, after the </InstanceProperties> tag.

   UP PvC adapter definition is displayed in Example 9-2, and can be found at:

   ```
   <install_path>\sg246171\adapter.pvc\UpPvcAdapter.xml
   ```

   WAP PvC adapter definition can be found at:

   ```
   <install_path>\sg246171\adapter.pvc\WapPvcAdapter.xml
   ```

Example 9-2 UP PvC adapter definition for the WCS instance XML config file

```
<HttpAdapters>
   <HttpAdapter
      name = "UP"
      deviceFormatId = "-2"
      deviceFormatType = "PVCAdapter"
      deviceFormatTypeId = "-1"
      factoryClassname="com.ibm.commerce.pvcadapter.up.UpPvcAdapter"
      enabled="true" >
      <PVCAdapter  registrationMode="0"
         preferredLogonTimeout="20"
         bufferTimeout="5" >
      </PVCAdapter>
   </HttpAdapter>
</HttpAdapters>
```

5. Start the WebSphere Commerce Server - <wcs_instance>.

9.3.6 Configure content management

The content management configuration refers to the database updates required for the device-specific content JSPs to be served to the mobile client. This is accomplished by executing SQL scripts to update the WCS database tables to set the document root for the device-specific content JSPs. Once the PvC adapter detects the device type of the request, the mobile device client is served the correct device-specific content JSPs based on the content management configuration.

To configure the content management, complete the following steps:

1. Start the DB2 command line processor by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command Window**.
2. Change to the directory of the content management script file.
   - For HDML content UP browser clients, do the following:
     i. Start a DB2 command line session.
     ii. Change to the directory of the script. For example, c:\temp\sg246171\direct\hdml\scripts.
     iii. Execute the content managment script by typing the following at the DB2 command line:

```
> db2 connect to <wcs_database>
> db2 -tvf setup_hdml.sql
```

   - For WAP WML content, do the following:
     i. Start a DB2 command line session.
     ii. Change to the directory of the script. For example, c:\temp\sg246171\direct\wap\scripts.
     iii. Execute the content managment script by typing the following at the DB2 command line:

```
> db2 connect to <wcs_database>
> db2 -tvf setup_wap.sql
```

Note: The settings in the setup_hdml.sql and setup_wml.sql have many of the same values. The settings will need to be modified to deploy and run both the UP, WAP PvC adapters and HDML, WML content simultaneously.

Information on modifying these settings can be found in the following sections:

- 11.3, “Deploying multiple PvC adapters” on page 222
- 12.1, “Content management configuration” on page 226
- “Content management reference” on page 364

3. When the database setup script is done processing, restart WebSphere Commerce Server - <wcs_instance> from the Administration Console, so that the changes can take effect.

9.3.7 Verify PvC Fashion sample store

We need to test the PvC Fashion sample store from a PC browser client and a mobile device client. In order to test the m-commerce direct PvC Fashion store, we will need to do the following:

1. Install the mobile device simulator (UP or WAP).
   - UP browser clients

     Refer to 14.1, “HDML toolkits and test clients” on page 266 for detailed instructions.
   - WAP browser clients

     Refer to 15.1, “WAP toolkits and test clients” on page 272 for detailed instructions.
2. From the simulator, enter the PvC Fashion store home page URL as noted and saved when launching the store for a PC browser client.

Important: If the URL starts with `https://` please replace it with `http://` , since the simulator we used for testing does not support SSL connection.

   - The UP HDML simulator should display the HDML content as seen in Figure 9-1.

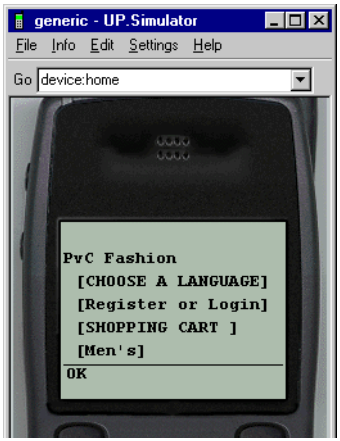


Figure 9-1 PvC Fashion samle store HDML content - UP HDML simulator

– Nokia WAP simulator should display the WML content as seen in Figure 9-2.

Figure 9-2 PvC Fashion sample store WAP WML content - Nokia WAP simulator

3. Verify the PvC Fashion sample store from a PC browser by entering the URL as noted and saved when launching the store for a PC browser client. The home page should look like Figure 9-3.

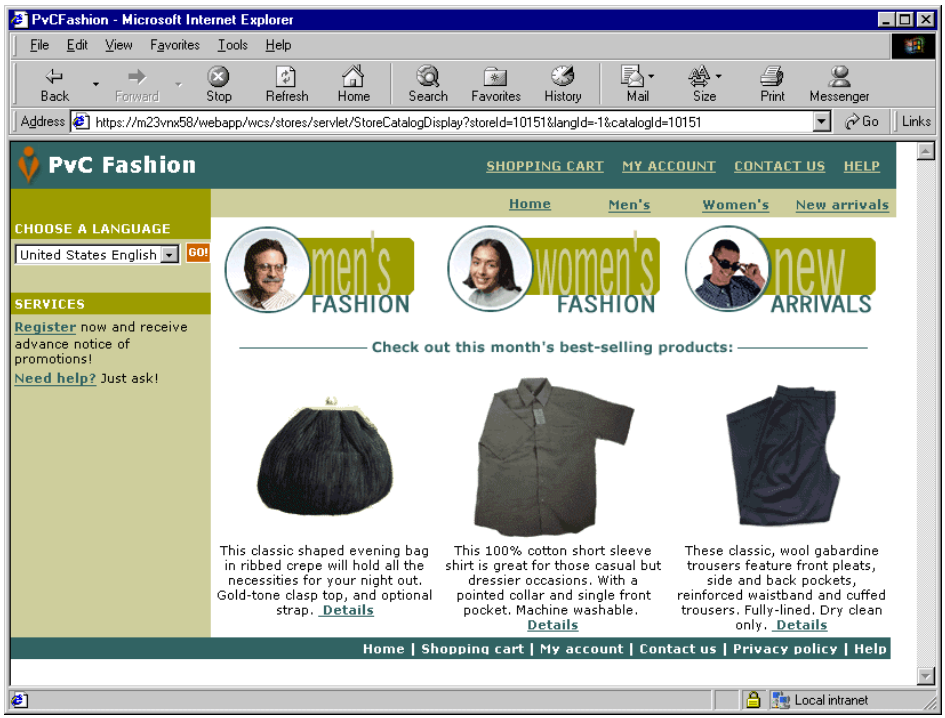


Figure 9-3 PvC Fashion sample store HTML content - PC browser client

9.4 Prepare the development environment

In preparation for development using the PvC Fashion store as a basis, we set up PvC Fashion to run in the VAJ WTE, and imported all the store assets into Studio. The PvC adapter and custom PvC commands are developed in VisualAge for Java. The device-specific content JSPs are developed in Studio.

We recommend that you follow the following procedure to prepare your development environment for the PvC Fashion store:

- Update VAJ WTE to run new store

  This is very beneficial for debugging and developing Java code.
- Load assets into Studio
- Load assets into VAJ

Refer to the *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167 for detailed information on completing the above-mentioned tasks.

Part 3

m-commerce direct implementation

10

m-commerce direct design and development process

This chapter describes application design guidelines for developing an m-commerce application using the direct approach, which includes device-specific content JSPs. In addition, we have included a summary of the development process.

The chapter is organized into the following sections:

- m-commerce direct application design guidelines
- m-commerce direct development process

10.1 m-commerce direct application design guidelines

The objective of this section is to describe how to take an existing store built for PC browser clients and enable the store for mobile device clients. We will highlight the key application design considerations.

Our application design provides guidance on what we did to enable the InFashion sample store for m-commerce. We have provided this modified version of InFashion, which we named PvCFashion, in the SG246171.zip.

In this section, we examine the following application design considerations:

- Content grouping based on device
- Creating content: copy PC JSPs for new mobile devices
- Modifying shopping flow
- Modifying content based on the limitations of the device
- WCS cache
- Dividing pages too large for display of device
- Converting content and images for target browsers
- Usability design
- Session management

Content grouping based on device

Decide how to group contents that you want to support for mobile devices of different capabilities and protocols. The JSPs will need to be developed and stored in different subdirectories. Through the use of the PvC adapter and configuration, the device will be detected and automatically use the appropriate JSPs with specific content for the device.

You must decide how to sort devices when using this function before modifying the JSP. The content creation workload will increase if your create very specific content groups based on the capability of the device. On the other hand, you can provide optimal performance with specific content for each type of mobile device. We recommend using common JSPs as much as possible to lessen the development workload and development cycle for a more rapid deployment.

Create the chosen JSP content-specific directory under the store directory. For example, the file system path (absolute) for our PvCFashion store is as follows:

```
c:\ibm\wcs\stores\web\PvCFashion
```

The PC browser client JSPs are in the root of PvCFashion. To distinguish between JSPs of the same name, we created a directory off the root of the store called hdml, for the HDML-specific content JSPs, as follows:

```
c:\ibm\wcs\stores\web\PvCFashion\hdml
```

Using the logonForm.jsp, here is how to configure the server:

- Client accesses the store
- The incoming HTTP request header is evaluated by the adapter to determine device type.
  - If a PC browser client us accessing the store, the server will search for the logonForm.jsp registered in VIEWREG table, for example:

    ```
    [Store directory]/logonForm.jsp]
    ```

    and process it to display the JSP output for the PC browser client.
  - If a mobile device (for example, an HDML-based client) is accessing the store, the server will search the VIEWREG table, for example:

    ```
    [Storedirectory]/hdml/logonForm.jsp]
    ```

    and process it to display the JSP output for HDML mobile clients.

The server configuration also includes registering data according to the device classification in PVCDEVMDL,PVCDEVSPEC and in the PVCMDLSPEC table. To achieve content switching and session control by subscriber ID (device type detection), the PvC adapter must be deployed.

Creating content: copy PC JSPs for new mobile devices

As a starting point, we recommend copying all the JSPs and images from the PC browser clients to develop the device-specific JSPs and convert the images. We also recommend that you import the copy of these JSPs into a folder created in Studio.

Modifying shopping flow

In our PvCFashion sample for mobile phones, we changed the shopping flow developed for PC browser clients to fit the needs and capability of the HDML mobile phone and user.

Modifying content based on the limitations of the device

In our example, we converted the HTML-based content to HDML and made modifications specific to the screen size of the target device (a five-line LCD screen).

We have listed some key limitations unique to mobile phones. We recommend checking for the following limitations based on the targeted mobile phone capabilities:

- If the target browser cannot support JavaScript, it is necessary to realize the source JavaScript in another way. When using WAP, convert the JavaScript to WMLScript.
- Certain pages that change dynamically, such as the search result of a product, may exceed the file size that the browser can receive. In this situation, you may need to divide the content into several pages.
- There is a limit in the length of a URL that can be sent at one time. We need to ensure that the URL length is not exceeded.

WCS cache

At the time of writing this redbook, it was a requirement that you disable the WCS cache. If you do not disable the cache, JSPs served from different directories with the same name cannot be distinguished by the WCS cache. The first JSP cached with a given name will potentially be loaded for clients of a different type.

For example:

- PC JSPs in root directory of store
- WAP WML JSPs in wml_jsp directory off the root of store

The StoreCategoryDisplay.jsp contains HTML for the PC version and WML for the WAP version. If the WAP client accesses the JSP first, it is compiled and cached. Next, the PC client accesses the store and will not get the cached version of the file instead of the correct version for the detected device type.

Dividing pages too large for display of device

When we send a large amount of data at once using the HTML <FORM> tag (for example for address registration), there is a possibility that it will exceed the capacity of the device memory. So we must divide the <FORM> tag into several pages. Here, the PvCBufferUrl command is used and several pages of data are saved on the server side, then processed by the server.

There are instances where page size changes dynamically by manipulation from the user side (for example for displaying search results, executing Category Display command, or displaying a shopping cart page). To be prepared for these eventualities, we divide the page based on the limitation of the devices.

Converting content and images for target browsers

The HTML-based JSPs need to be modified for the target browser's markup language of the mobile device. For example, we modified the HTML content to HDML and converted the images. Image files are often made smaller and the resolution is lowered so that it will not be beyond the capacity of the mobile device.

Usability design

The design of the user interface needs to be carefully considered and tested using software-based simulators and real mobile device hardware.

- Simulator

  After the conversion process, the JSP files will be tested using a simulator for the mobile device. This test shows whether the display flow is properly working.

- Mobile device hardware

  After testing with simulators, we need to test with as many types of mobile hardware devices as possible. Each device has a different displayable screen size, so this test is very important. Browsers that are loaded on the mobile phone devices sometimes differ, depending on the manufacturers of the mobile phones.

  Often, the simulator and actual hardware do not produce the same result. This is due in part to the simulator features not keeping pace with the features of mobile device hardware. Because of these differences, the same page may be displayed differently. There are browser software bugs that sometime impede the proper display of pages. The same JSP behaves differently on different mobile device browsers. To avoid surprises, we recommend you test on as many mobile hardware devices as possible.

Session management

When developing the PvC adapter, you will need to understand the behavior of the mobile device for session control. Refer to 3.1.2, "Session management" on page 72 for more information.

10.2 m-commerce direct development process

Before we get started developing the m-commerce application, we need to verify that the following prerequisites have been met.

1. Runtime environment

   The WCS V5.1 runtime environment is set up with a WCS instance.

Refer to Chapter 7, “m-commerce runtime environment” on page 141 for details.

2. Deploy the PvC Fashion sample store

   The sample store is deployed and working.

   Refer to Chapter 9, “m-commerce sample store and sample code” on page 179 for details.

3. Development environment

   The development environment is installed and working (Studio, VisualAge for Java, WTE, XML editor, etc.).

   Refer to Chapter 8, “m-commerce development environment” on page 159

4. Mobile device simulator

   For development unit testing, we need to ensure that we have a test simulator for the desired wireless protocol installed and configured.

   – UP browser HDML content

     In the examples to follow in this chapter, we need to install the simulator based on UP.SDK V3.2 for HDML.

     Refer to Chapter 14, “HDML implementation sample” on page 265 for detailed installation and configuration instructions.

   – WAP browser WML content

     Refer to Chapter 15, “WAP implementation sample” on page 271 for detailed installation and configuration instructions.

   – Palm HTML content

     Refer to Chapter 16, “Palm implementation sample” on page 279 for detailed installation and configuration instructions.

5. Create a PvC adapter
6. Deploy a PvC adapter
7. Content management configuration
8. Create device-specific content JSPs
9. Deploy content
10. Create custom PvC commands
11. Test the m-commerce store

11

Creating and deploying a PvC adapter

This chapter describes development procedures for creating and deploying a PvC adapter. In this section we provide instructions for creating a carrier-specific PvC adapter. The information provided applies to creating adapters for any wireless protocol.

This chapter is organized into the following sections:

- Creating a PvC adapter
- Deploying a PvC adapter
- Deploying multiple PvC adapters

11.1 Creating a PvC adapter

This section provides detailed information on how to create a PvC adapter. Depending on your preferences, you can develop the adapter from the following procedure or import sample PvC adapter. In this chapter we will demonstrate how to create a PvC adapter for a UP carrier -specific browser. As noted below, we have included sample PvC adapter code for UP, WAP, Palm, and WTP.

We have organized the steps required to create a PvC adapter as follows:

- Create a project in VisualAge for Java (VAJ)
- Create a package in VAJ
- Identify the device type
- Create an adapter class
- The checkDeviceFormat method
- The getDeviceModel method
- The getTerminalId method

PvC adapter sample code

The sample code described in this section can be found in the additional materials .zip file, SG246171.zip, in the following directory after being unzipped:

```
<install_path>\sg246171\adapter.pvc
```

Each of the following samples includes the Java class and methods found in the JAR file, which can be imported into VAJ, and the PvC adapter definition XML file used to update the WCS instance XML file for deployment.

Note: The PvC adapter JAR files include the Java source and class. When deploying the adapter on a production system, we recommend that you export the JAR without the Java source.

We have included sample PvC adapters for the following:

- For UP HDML
  - UpPvcAdapter.jar
  - UpPvcAdapter.xml

  This PvC adapter is for use with UP.Browser (HDML) mobile devices. This code is used throughout this chapter as a working example of how to develop a PvC adapter.

Refer to Chapter 14, “HDML implementation sample” on page 265 for more detailed information.

- For WAP
  - WapPvcAdapter.jar
  - WapPvcAdapter.xml

  This PvC adapter is for use with WAP mobile devices with browser support for WML content.

  Refer to Chapter 15, “WAP implementation sample” on page 271 for more detailed information.
- For WTP
  - WapPvcAdapter.jar
  - WapPvcAdapter.xml

  Refer to Chapter 19, “m-commerce using WTP application development” on page 305 for more detailed information.

11.1.1 Create a project in VisualAge for Java (VAJ)

Before we create the PvC adapter, we need to create a project in VAJ by completing the following steps:

1. Start VisualAge for Java.
2. Select **File** -> **Quick Start** from the VAJ main menu.
3. When the Quick Start window appears, select **Basic** -> **Create Project**, then click **OK**.
4. When the Add Project window appears, select **Create a new project named:**, then enter the project name. For example, we entered `PvC Adapter for UP HDML`. Then clicked **Finish**.

Note: The VAJ project name could be anything you like. We have provided a specific name to make it easier to refer to throughout the instructions.

11.1.2 Create a package in VAJ

This section describes the steps necessary to create a package in VAJ, which will be used in creating an adapter. In this example, we will create a package called com.ibm.commerce.pvcadapter.hdml by completing the following steps:

1. Select the project created in the previous section.
2. Right-click **Add,** then click **Package**.

3. When the Add Package window appears, enter the new package name: `com.ibm.commerce.pvcadapter.up` and then click **Finish**.

11.1.3 Identify the device type

Before creating the adapter, we need to identify the device type we want to check for in the adapter. We have provided a Java utility (see Example 11-1) called PrintHttpRequest.java to help us determine the user-agent action to look for. When a request is transmitted from the mobile client browser, the PrintHttpRequest application will wait for the connection on the port configured (for example, port 9998). When the client connects, a message is transmitted to standard output.

Example 11-1 PrintHttpRequest.java

```
import java.io.*;
import java.net.*;
class PrintHttpRequest {
   public static void main (String args[]) {
      ServerSocket serverSocket = null;
      Socket socket = null;
      try {
         serverSocket = new java.net.ServerSocket(9999);
         socket = serverSocket.accept();
         BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
         String line = "";
         while (line != null) {
            line = reader.readLine();
            System.out.println(line);
         }
      } catch (Exception e1) {
         e1.printStackTrace();
      } finally {
         try {
            if (socket != null) socket.close();
            if (serverSocket != null) serverSocket.close();
         } catch (Exception e2) {
            e2.printStackTrace();
         }
      }
   }
}
```

Import PrintHttpRequest into VAJ

Prior to executing PrintHttpRequest.java, it must be imported into VAJ by completing the following steps:

1. Open VisualAge for Java.
2. Select the project created in the previous section (for example, PvC Adapter for UP HDML).
3. Select **File** -> **Import**. Select the import source **Jar**, click **Next**.
4. When the Import from a directory window appears, do the following:
   – Click **Browse** to select the directory and file, for example c:\sg246171\adapter.pvc\PrintHttpRequest.jar and then click **Open**.
   – Select the type of file to import: **.java**. Click **Details** next to .java, and select **PrintHttpRequest.jar**. Click **OK**.
   – Deselect **Resource**.
   – Click **Finish**

You should now see a new package called com.ibm.commerce.pvcadapter.utils containing PrintHttpRequest.

Run PrintHttpRequest in VAJ

To run PrintHttpRequest in VAJ, complete the following steps:

1. Select **Main** in the PrintHttpRequest class.
2. Change the ServerSocket (9999) to a port that is not in use. For example we changed this port to 9998.
3. To run, select **Main**, right-click **Run**, then select **Run Main**.
4. From your simulator, enter the following URL to access your PrintHttpRequest test program:

   ```
   http://localhost:9998/
   ```

Note: In this example we used the UP.SDK V3.2 for HDML. For more information on this SDK and simulator, refer to 14.1, “HDML toolkits and test clients” on page 266.

5. The VAJ Console should display output as seen in Example 11-2. Take note of the following HTTP header information:
   – Accept: <value>
   – User-Agent: <value>

Example 11-2 VAJ Console output from PrintHttpRequest

```
GET / HTTP/1.1
Accept-Charset: ISO-8859-1
Accept-Language: en
Content-Type: application/x-www-form-urlencoded
```

```
x-up-subno: admin_jganci.itso.ral.ibm.com
x-upfax-accepts: none
x-up-uplink: none
x-up-devcap-smartdialing: 1
x-up-devcap-screendepth: 1
x-up-devcap-iscolor: 0
x-up-devcap-immed-alert: 1
x-up-devcap-numsoftkeys: 2
x-up-devcap-screenpixels: 171,108
x-up-devcap-msize: 8,18
Accept: application/x-hdmlc, application/x-up-alert,
application/x-up-cacheop, application/x-up-device,
application/x-up-digestentry, text/x-hdml;version=3.1,
text/x-hdml;version=3.0, text/x-hdml;version=2.0, text/x-wap.wml,
text/vnd.wap.wml, */*, image/bmp, text/html
User-Agent: UP.Browser/3.1-UPG1 UP.Link/3.2
Host: localhost:9998
```

In this example, HTTP header User-Agent tells us that this browser is a UP.Browser/3.1-UPG1 UP.Link/3.2 and the Action specifies the type of content supported by the browser. We will use this information when creating the adapter in subsequent steps.

Note: You will need to go through this process to identify each device type user-agent. When creating the adapter in subsequent steps, you must explicitly check for the user-agent of the device.

11.1.4 Create an adapter class

In this section, we create an adapter class that extends the superclass com.ibm.commerce.pvcadapter.PVCAdapterImpl.

Create class

To create the class, complete the following steps:

1. Select the **com.ibm.commerce.pvcadapter.up** package under the PvC Adapter for UP HDML project.
2. Right-click **Add**, then click **Class**.
3. When the Create Class window appears, as seen in Figure 11-1:
   - In the Class name field, enter UpPvcAdapter
   - Click **Browse** next to the Superclass field and enter `PVC` in the pattern field.
   - Select **PVCAdapterImpl** from the list of Type Names.

– Click **OK**.
– Click **Finish**.

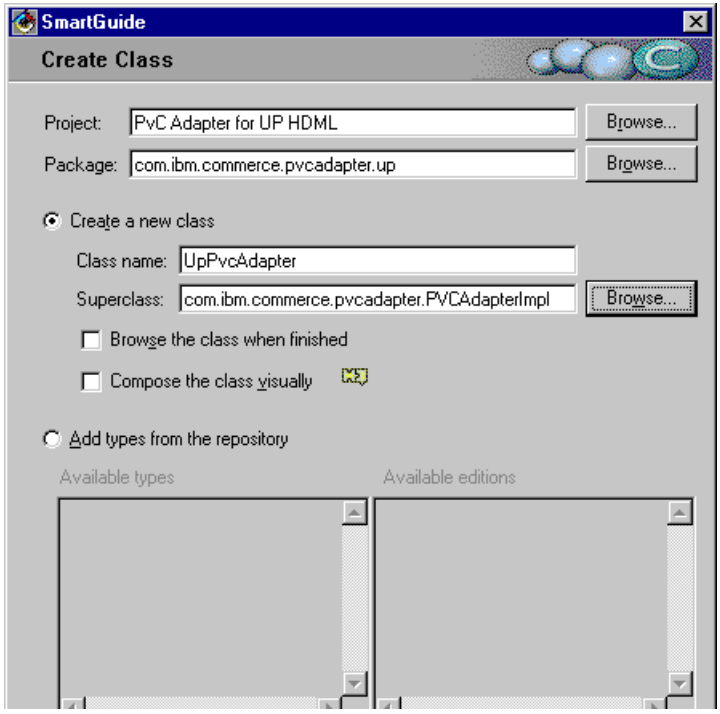


Figure 11-1 Creating the UpPvcAdapter class

4. After the class is created, the class should look like Figure 11-2 in VAJ.

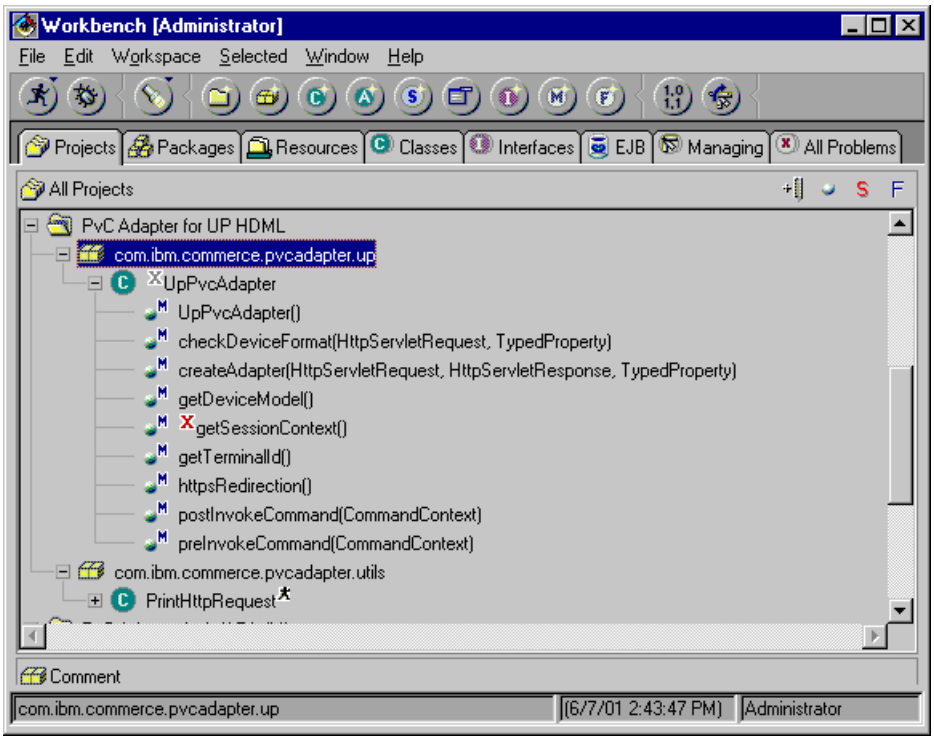


Figure 11-2 UpPvcAdapter class after creation

Removing unnecessary methods from the class

When creating a new adapter class using the wcs5101.dat from WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000, some unnecessary methods will get created. To remove the unnecessary methods, complete the following steps:

1. Select the **UpPvcAdapter** class found under the com.ibm.commerce.pvcadapter.samples package.
2. Select the following methods, and right-click **Delete**. Click **Yes** to confirm each delete.
   - createAdapter
   - getSessionContext
   - httpsRedirection
   - postInvokeCommand
   - preInvokeCommand

Important: The following methods are required, and should not be deleted:

- checkDeviceFormat
- getDeviceModel
- getTerminalId

After removing the unnecessary methods your class should look like Figure 11-3.

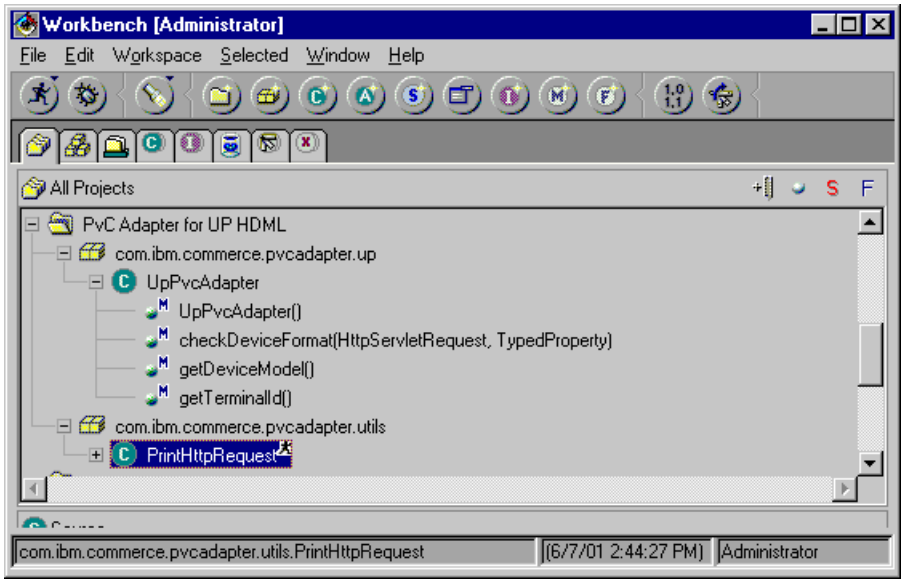


Figure 11-3 UpPvcAdapter class after removing unnecessary methods

11.1.5 The checkDeviceFormat method

This section provides a description of the checkDeviceFormat method and intended functionality. It also shows a procedure for creating the method.

Description of checkDeviceFormat method

The checkDeviceFormat method provides the following functionality:

- Checking if the request header contains a subscriber ID in the x-up-subno field
- Checking the format of the subscriber ID
- Comparing the host name in the subscriber ID with the name of the remote host

The WCS V5.1 product architecture can detect which adapter should be used to process a client request.

There are two steps in this method that are repeated until a suitable adapter is found. If a suitable adapter is not found, the session will be handled as if accessed from PC browser client.

First, the WCS server checks an adapter to determine if the request contains the correct information in the request from the carrier that the PvC adapter supports.

Second, the WCS server checks if the address of the sender is valid for the chosen adapter by comparing it with the address of the host listed in the configuration file, or with the network address.

If you know the address of a gateway, you do not need a description of this logic when an adapter is made, since an IP address check is done by setting the XML configuration file. Each adapter needs to include the checkDeviceFormat logic that returns the result, to determine whether it is a request that can be processed by the adapter.

The contents of the request from the browser are an object from the HttpServletRequest class, and a request parameter as an object of the TypedProperty class. We can determine if the request can be processed by the PvC adapter by examining the contents of the object delivered as an argument of the checkDeviceFormat. Although an IP check can be done by configuration by specifying a list of gateway addresses, in the case of the UP.Simulator, the name of the gateway host is present in the HTTP request as a part of subscriber ID.

We can also write extra logic that will check if the host name in the subscriber ID is the same as the name of the remote host. In this case you do not need a list of gateway IP addresses in the XML configuration file. The subscriber ID generated by the gateway by checking the host name in the subscriber ID and the host name of the requester will be the verification for this check. In other words, for UPLinkGateway simulated by UP.Simulator, you do not need to know all possible addresses of the UPLinkGateway for the following reasons:

- A subscriber ID from UPLinkGateway is created to verify that the subscriber ID is really sent by the same host as in the subscriber ID.
- UPLinkGateway is responsible for validating the subscriber ID sent by itself.

Note: This sample uses the UP.Simulator (Up.Link Gateway). Regarding format of the subscriber ID for actual gateway please refer to the manual provided by the carrier.

Create the checkDeviceFormat method

In checkDeviceFormat method we will perform the following checks, as seen in Example 11-3:

- Check if request header contains subscriber ID in the x-up-subno field
- Check the format of the subscriber ID
- Compare the host name in the subscriber ID with the name of the remote host

Example 11-3 checkDeviceFormat method - PvC Adapter

```
public boolean checkDeviceFormat(
      javax.servlet.http.HttpServletRequest arg1,
      com.ibm.commerce.datatype.TypedProperty arg2)
{
   String subscriberId = arg1.getHeader("x-up-subno");
   if (subscriberId == null) return false;
   int i = subscriberId.indexOf("_");
   if ( ( i<=0 ) || ( i>=subscriberId.length() ) ) return false;
   String host = subscriberId.substring(i+1);
   String remoteIP = arg1.getRemoteAddr();
   try {
      String gwIP = java.net.InetAddress.getByName(host).getHostAddress();
      return remoteIP.equals(gwIP);
   } catch (java.net.UnknownHostException e) {
      return false;
   } catch (NullPointerException e) {
      return false;
   }
}
```

Notes to Example 11-3:

- The first line in the body of the method reads the subscriber ID from the request. The value of the header fields are acquired by the getHeader method.
- Checks if the subscriber ID is contained in the request by checking whether return value is null or not.
- Gets the index of separator, which separates the user ID and the host name of the machine executing UP.Simulator.
- Checks if the separator is found and is located in the correct place.
- Checks if the host name in the subscriber ID and the host name of the requester is the same. If it is not the same, it returns false and assumes that the access from the client is not valid.

  For example:

  Subscriber ID: `userid_m23bk61w.itso.ral.ibm.com`

Sent from the host: m23bk61w.itso.ral.ibm.com

It returns true because the request is sent from a valid address. If it fails, it means that someone has stolen the subscriber ID and is trying to send the same subscriber ID from another location by using a different type of device, such as a PC. Thus, by checking the client address and host name in the subscriber ID, the server can block illegal access from other locations.

Note: This kind of invalid access from the same gateway should be blocked by the gateway itself.

11.1.6 The getDeviceModel method

In order to choose the proper content for a targeted device, it is essential that we determine the type of device. This is done using the getDeviceModel method.

How to determine the type of mobile device

The WCS server can extract the model name from an adapter when trying to select suitable content for a requesting device. How the name of the device is stored depends on the service. The WCS server expects data returned by the getDeviceModel method from an adapter. Sometimes the model name of the device is set as a substring of the header field. The model name of a device cannot be extracted from a request in the same way. Therefore, each adapter needs to implement getDeviceModel to extract the model name of the accessing device.

Creating the getDeviceModel method

In the case of the UPLinkGatewayAdapter, the model name is found between '-' and the next blank character of the second element of the User-Agent field delimited by a slash. We describe a program that extracts the part from User-Agent field and returns it in the getDeviceModel method. To write the code that accesses a client request, you can obtain the contents of the request by the getRequest() method defined in the PVCAdapterImpl of the superclass. Example 11-4 is the program for a UPLinkGatewayAdapter that retrieves the model name from a request.

Example 11-4 getDeviceModel method - UP PvC adapter

```
public String getDeviceModel() {
   String agent = getRequest().getHeader("User-Agent");
   java.util.StringTokenizer st = new StringTokenizer(agent);
   if (st.hasMoreTokens()) st.nextToken(); else return “”;
   if (st.hasMoreTokens()) {
      String model = st.nextToken();
      int begin = model.indexOf("-");
      int end = model.indexOf(" ");
```

```
            if ((begin <= 0) || (end <=0)) return "";
            if ((begin > end) || (end >= model.length())) return "";
            model = model.substring(begin+1,end);
            return model;
         } else {
            return "";
         }
      }
```

Important: Don't return null when you get an unexpected value in the user agent field. In this case, return an empty string."" as the default.

In order to use the StringTokenizer, you will need to add the following to the UpPvcAdapter class at the top of the file:

```
import java.util.*;
```

The code is broken down as follows:

- Extracts the value of the User-Agent field by the getRequest method.
- Using the StringTokenizer, the value of the User-Agent field is divided with "/", and the second element is parsed into the String object called model.
- The string between '-' character and the next blank is extracted and is returned as the model name.

11.1.7 The getTerminalId method

This section explains how to determine the identification information and provides an example of the getTerminalId method.

How to determine identification information

The subscriber ID is an element that is indispensable for session control. The WCS server expects adapters to return the subscriber ID as a return value of the getTerminalId method. Returning the value as the getTerminalId allows for session control by the adapter. In the case of the UP PvC adapter, the subscriber ID is set in the header field as `x-up-subno` (see Example 11-5). The logic to retrieve this value has been provided in getTerminalId as seen in Example 11-5.

Example 11-5 getTerminalId method - UP PvC adapter

```
public String getTerminalId() {
   return getRequest().getHeader("x-up-subno");
}
```

It is possible to retrieve the request from the browser as an object of the HttpServletRequest class using the getRequest() method. This method retrieves the value of x-up-subno from the request from the browser extracted by the getRequest() method. Since the correct value is confirmed by the checkDeviceFormat method, we only retrieve the value.

Naming the adapter

When providing services to devices of several carriers, there is a possibility that the subscriber ID that a service provider generates is the same as another service provider. In this event, even if the subscriber ID is obtained, it cannot be used to distinguish the subscriber ID from another carrier. To avoid such a problem, WCS V5.1 provides a method to assign a unique ID to every device.

Because carriers could potentially send the same ID, we create a unique ID with a combination of the names given to the adapter and to the subscriber ID, for example, the name of the adapter for business person A is 'serviceA' and the name of the adapter for business person B is 'serviceB'. Even when receiving the same ID '001122334455667788' from different mobile phones and adapters, so far as the adapter name is unique, the combination of the adapter name and the subscriber ID will always be unique, as follows:

- ('serviceA','001122334455667788')
- ('serviceB','001122334455667788')

WCS V5.1 specifies the device based on the combination of the adapter and the subscriber ID.

Also, a unique name is needed that will not overlap that of other adapters when creating a new adapter. The combination of an adapter name and a subscriber ID will always be unique.

11.2 Deploying a PvC adapter

The steps in deploying a PvC adapter to your m-Commerce WCS V5.1 runtime environment are as follows:

- Exporting the PvC adapter from VAJ to a JAR file
- Deploying the PvC adapter JAR to the WCS server
- Adding a PvC adapter definition to the WCS instance XML file

11.2.1 Exporting the PvC adapter from VAJ to a JAR file

To export the PvC adapter from VAJ to a JAR file to be deployed in the WCS V5.1 runtime, complete the following steps:

1. From the VAJ Workbench, select the PvC adapter package.

   For example, we selected the **com.ibm.commerce.pvcadapter.up** package in the PvC adapter for UP HDML project.

2. Right-click and select **Export**.
3. When the Export window appears, select the **Jar** file under the heading export destination, and then click **Next**.
4. When the Export to a JAR file window appears, do the following:
   - Enter the path and file name in the Java file field: `c:\temp\UpPvcAdapter.jar`
   - Select **.class**
   - Deselect **.java**

     When packaging the JAR for deployment, we do not want to include the Java source code.
   - Click **Finish**

11.2.2 Deploying the PvC adapter JAR to the WCS server

Now that we have created the JAR file for the PvC adapter, we need to install and configure the JAR on WebSphere Application Server to make the adapter accessible from a WCS request servlet.

To install and configure the adapter on the WCS server, complete the following steps:

1. Copy the PvC adapter JAR file to the <wcs_install_path>\lib directory.

   For example, copy `UpPvcAdapter.jar` from `c:\temp` to `c:\ibm\wcs\lib`.

2. Add the PvC adapter JAR file to the WAS class path.
   a. Start the WebSphere Application Server Administrator's Console.
   b. Expand the **WebSphere Administrative Domain** and select the hostname node.
   c. Right-click **WebSphere Commerce Server** and select **<instance_name>**.
   d. Click **Stop**. After a few moments, a message is displayed stating that it stopped successfully.

e. In the Command line arguments field, enter the path and file name of the PvC adapter JAR at the end of the class path parameter. For example:

```
<wcs_install_path>\lib\UpPvcAdapter.jar
```

f. Click **Apply** to apply the changes.

g. Right-click **WebSphere Commerce Server** and click **<instance_name>**.

h. Click **Start**. After a few moments, a message is displayed stating that it started successfully. You can now close the administrator's console.

11.2.3 Adding a PvC adapter definition to the WCS instance XML file

In order for WCS to know of the existence of the PvC adapter, we must add the PvC adapter definition to the WCS instance XML configuration file. The definition opens with the tag <HttpAdapters> and ends with the tag </HttpAdapters>.

To add the PvC adapter definition to the WCS instance XML configuration file, complete the following steps:

1. Stop the WCS Server.
2. Change to the <wcs_install_path>\instances\<instance_name>\xml directory.

   For example, c:\ibm\wcs\instances\wcs\xml
3. Copy the <wcs_instance>.xml file to <wcs_instance>.xml.bak.
4. Open the <wcs_instance>.xml with an editor. Add the PvC adapter definition to the end of the file. Here is a template for the PvC adapter definition for WCS instance XML file.

```
<HttpAdapter [HTTP Adapter Attribute] >
         <PVCAdapter [PVC Adapter Attribute]>
      <IPCheck>
         <IP [IP Attribute 1]>
         ...
         <IP [IP Attribute n]>
      </IPCheck>
      <ExcludeCommands>
         <Command name=[[Excluded Command Name 1]/>
         ...
         <Command name=[Excluded Command Name n]/>
      </ExcludeCommands>
      <RelogonCommands>
         <Command name=[Protected Command Name 1]/>
         ...
         <Command name=[Protected Command Name n]/>
      </RelogonCommands>
   </PVCAdapter>
</HttpAdapter>
```

5. We modified the following fields as seen in Example 11-6.
   - [HTTP Adapter Attribute]
   - [IP Adapter Attribute 1]
   - [IP Attribute n]
   - [Excluded Command name 1]
   - [Excluded Command Name n]
   - [Protected Command Name 1]
   - [Protected Command Name n]

Note: For details on all the parameters, refer to “PvC adapter definition for the WCS instance XML file” on page 330.

Example 11-6 Example: PvC Adapter definition for WCS instance XML file

```
<HttpAdapters>
<HttpAdapter
   name = "UPG"
   deviceFormatId = "-2"
   deviceFormatType = "PVCDevice"
   deviceFormatTypeId = "1"
   factoryClassname="com.ibm.commerce.pvcadapter.up"
   enabled="true" >
   <PVCAdapter
   registrationMode="1"
   preferredLogonTimeout="60"
   bufferTimeout="10" >
      <IPCheck>
         <IP type="net" value="192.168.0.0" mask="24" />
         <IP type="host" value="9.168.91.113"/>
      </IPCheck >
      <ExcludeCommands>
         <Command name="AddressAdd"/>
         <Command name="AddressDelete"/>
      </ExcludeCommands>
      <RelogonCommands>
         <Command name="OrderProcess"/>
      </RelogonCommands>
   </PVCAdapter>
</HttpAdapter>
</HttpAdapters>
```

11.3 Deploying multiple PvC adapters

The deployment of more than one PvC adapter simultaneously requires that the adapter definition and the content management scripts be modified to provide unique values for the configuration settings.

11.3.1 PvC adapter definition

When deploying more than one PvC adapter, we recommend that you configure the adapter definition with a unique value for the deviceFormatId as seen in Table 11-1.

Table 11-1 PvC adapter definition for multiple PvC adapters

Adapter definition	UpPvcAdapter value	WapPvcAdapter value
deviceFormatId	‘1’	‘2’
deviceFormatTypeId	‘-1’	‘-1’

Some general guidelines for setting the values for deviceFormatId and deviceFormatTypeID are as follows:

- deviceFormatId

  The main purpose of this element is to identify the unique adapter.

  A unique integer value must be set for each adapter. For example, in Table 11-1 the deviceFormatId value for the UpPvcAdpater is ‘1’ and the WapPvcAdapter is ‘2’.

- deviceFormatTypeId

  The value of the deviceFormatTypeId is used by the WCS server to search for JSPs in the view registry. We recommend that you set the deviceFormatTypeId value to ‘-1’.

  If you specify a different value, you need to register views in the VIEWREG, DISPCGPREL, and DISPENTREL tables for the DEVICEFMT_ID value.

These elements have different roles. The deviceFormatId is used for the purposes of identifying each adapter, and deviceFormatTypeId is used for controlling VIEW file that will be sent for the client.

If you set the deviceFormatId to '2', the server tries to search the view that is stored in VIEWREG, DISPCGPREL or DISPENTRELI for DEVICEFMT_ID equal to '2' prior to the view DEVICEFMT_ID equal to '-1'.

If no view is found with specified deviceFormatTypeId, then the server tries to search the view for DEVICEFMT_ID value equal to '-1'. The value '-1' is defined for PC browser clients, and is listed in the DEVICEFMT table. By specifying '-1' for deviceFormatTypeId, we share entries in the VIEWREG table between the PC browser client and the mobile device client.

By specifying '2' for deviceFormatTypeId, we provide a device-specific view for the adapter with the value '2' for DEVICEFMT_ID. If there is no device-specific view in the VIEWREG, DISPCGPREL or DISPENTRELI tables, we use the view for a PC browser. However, this only works for views in the VIEWREG table.

Restriction: At the time of writing this redbook, the catalog subsystem was not able to support multiple type of devices. This problem has been reported.

If you specify deviceFormatTypeId to other value than '-1', catalog page will not send to mobile client because of the problem.

As a workaround for this problem, copy all entries in DISPCGPREL and DISPENTRELI table with DEVICEFMT_ID you specified.

In summary, we recommend that you always specify '-1' for deviceFormatTypeId for the PvC adapter definition.

11.3.2 Content management configuration

When using two or more PvC adapters simultaneously, the content management configuration will need to be modified to include unique values for the model_id, mdlspec_id, and spec_id.

For example, if you are deploying a UpPvcAdapter and a WapPvcAdapter and want mobile devices for each to access your server m-commerce store, refer to the settings in Table 11-2.

When you change model_id and spec_id, you need to update the relationship between these keys that will be stored in the PVCMDLSPEC table.

Table 11-2 Content management configuration for multiple PvC adapters

PvC database table elements	UpPvcAdapter content management sql script value	WapPvcAdapter content management sql script value
model_id	-1	-20
mdlspec_id	100	200
spec_id	100	200

Example 11-7 includes a sample of the SQL script containing the values defined in Table 11-2 for the WAP content management configuration. The contents of Example 11-7 can be found at:

```
<install_path>\SG246171\direct\wap\scripts\setup_wap1.sql
```

The UP HDML content management values are the same as defined in our sample script setup_hdml.sql.

Example 11-7 Content management setup_wap1.sql for multiple PvC adapters

```
insert into pvcdevmdl (
   MODEL_ID,
   MODELNAME,
   SESSIONTYPE,
   VENDOR,
   DESCRIPTION
) values (
   -2,
   '',
   'WAP',
   '',
   ''
);

insert into PVCDEVSPEC (
   SPEC_ID,
   SPECNAME,
   SESSIONTYPE,
   CONTENTDIR
) values (
   200,
   'WAP Default',
   'WAP',
   'wml_jsp'
);

insert into pvcmdlspec (
   MDLSPC_ID,
   STOREENT_ID,
   MODEL_ID,
   SPEC_ID
) values (
   200,
   0,
   -2,
   200
);
```

12

Create, deploy and manage content

When developing an m-commerce application for the direct approach, you need to develop device-specific content JSPs. For example, if you are targeting WAP mobile phone, you need to develop JSPs with WAP WML content. In this chapter, we describe how to create device-specific content, deploy it, and perform the content management configuration of the WCS database tables.

This chapter is organized into the following sections:

- Content management configuration
- Create device-specific content JSPs
- Deploy content
- Advanced content and configuration example

12.1 Content management configuration

Content management includes updates required by the WCS database tables to set settings such as the content directory of the device-specific JSPs, and information about the device extracted by the PvC adapter. Once the device type is known by the information extracted from the HTTP request by the PvC adapter, the content directory for device-specific JSPs can be set. This configuration allows us to set the root directory for all JSPs without having to update the VIEWREG table for each JSP.

In summary, for all devices whose connections are handled by a PvC adapter, content management provides the following:

- Auto content selection of device-specific JSPs
- User device spec information settings as attributes to JSP programs

12.1.1 Content management

This section includes a description of the PvC tables used to configure content management for m-commerce.

Refer to the Appendix D, “PvC database tables and content management reference” on page 359 for more detailed reference information.

WCS PvC database tables

All of the necessary configuration changes are contained in three PvC database tables:

- PVCDEVMDL

  This table stores model information for mobile devices.

- PVCMDLSPEC

  This table stores specifications information and the directory of contents. One record can be shared by more than two devices.

- PVCDEVSPEC

  This table stores the relationship between model and spec. By using this table, device models which have similar specifications can be categorized in one record in the PVCDEVSPEC table.

In addition to the tables used for content management configuration, WCS V5.1 includes the following PvC database tables used for session management and URL buffering. These tables do not need to be configured.

- PVCBinding
- PVCSession
- PVCBuffer

In order for the JSPs to be displayed on the mobile device, we must perform the content management configuration to update the WCS database and deploy the JSPs to the runtime server in a unique directory. When a mobile client accesses an m-commerce store using WCS V5.1, the HTTP request is read by the WCS server. Next the PvC adapter determines the device type from the HTTP header and sets the content directory to serve the proper version of the JSP for display on the mobile device.

How does the content management work?

When a request comes from a browser, the server extracts the model name from the PvC adapter. The server then looks for the record related to the extracted device model and adapter name. If a suitable record for the client's device is found in the PVCDEVMDL table, using the relationship stored in PVCMDLSPEC table, a suitable device-specification can be found in PVCMDLSPEC.

The server will use the value of PVCMDLSPEC.CONTENTDIR in the selected record for content switching or setting the directory to the device-specific JSPs. If there's no mapping between the selected and a record in the PVCMDLSPEC table, the server tries the same search by using a record in the PVCDEVSPEC table whose MODELNAME is blank and SESSIONTYPE is the same as the adapter name. This kind of model record is called the adapter default model. We recommend that you prepare this adapter default.

The PVCMDLSPEC table has a STOREENT_ID column, so that you can define a different mapping from PVCDEVMDL to the PVCDEVSPEC table in the PVCMDLSPEC table for each store. If you set STOREENT_ID to 0, the record will be the site default setting. If there were defaults for the site-specific and store-specific settings at the same time, the site-specific mapping is used prior to the site default setting. In other words, the most suitable record in PVCDEVSPEC table is selected with the following priority:

1. Use store-specific mapping prior to site default record.
2. Use model-specific mapping prior to model default record.

Finally, the most suitable record is found in the PVCDEVSPEC table. The selected record is used for a contents switch and terminal specification reflection for the JSPs. If there are no suitable records in the PVCDEVSPEC table, the view commands act the same as an access from a PC browser client.

Note: If you do not need to change the JSP root directory for the created PvC adapter and you do not need to use the data stored in the PVCDEVSPEC table, you do not need to set up the database for the content management function and can skip the following section.

Define the adapter default model

First, we need to set up the PVCDEVMDL table for all devices whose session is managed by the PvC adapter.

Note: At the time of writing this redbook, all mobile devices that use the PvC adapter and content management functionality have sessions managed by WCS, regardless of the mobile device or gateway support for session management.

For example, we will define that all mobile devices detected by the UP PvC adapter will browse JSPs from the hdml_jsp content directory. This directory contains JSPs of the same name as the JSPs in the root used for PC browser clients. To represent the adapter common setting, we need to prepare one record that has a blank MODELNAME. The SESSIONTYPE of the record should be the same as the adapter name. A blank record in MODELNAME and SESSIONTYPE is the same as the adapter name, and referred to as the adapter default model.

When a model that is not listed in the PVCDEVMDL table is found, the server will automatically insert a new record into the PVCDEVMDL table. If there is no specific model-spec mapping in the PVCMDLSPEC table, the adapter default model will be used in searching for a suitable record in PVCDEVSPEC.

The following is a sample SQL script for inserting a record into the PVCDEVMDL table so as to set the default model for the UP PvC adapter:

```
insert into PVCDEVMDL (
   MODEL_ID,
   MODELNAME,
   SESSIONTYPE,
   VENDOR,
   DESCRIPTION
) values(
   -1,
   '',
   'UP',
   'Default',
   'Default model for UpPvcAdapter'
);
```

Auto content selection

The auto content selection uses the value of the CONTENTDIR in a detected record in the PVCDEVSPEC table. If the server succeeds in selecting a record from the PVCDEVSPEC table, it inserts the value of the CONTENTDIR to the search path of the JSPs, to be used for display on the mobile device.

The search-and-insert function works in either of the following cases:

- The request contains a store ID and does not have the option “storeDir=no” in the request URL. For example:

  <STORE_DIRECTORY>\<VALUE_OF_CONTENTDIR>\<JSP_FILE_PATH>

- The request does not contain the store ID or has the option “storeDir=no” in the request URL. For example:

  <VALUE_OF_CONTENTDIR>\<JSP_FILE_PATH>

From the JSP file, the value of the selected record in PVCDEVSPEC table is accessible through an object of the DeviceInfo class, which is set as an attribute of the HTTP request for the JSP file.

Example 12-1 shows a sample SQL script for inserting a record into the PVCDEVSPEC table in order to set the CONTENTDIR to hdml_jsp, and the SESSIONTYPE to the adapter name UP.

If you already have records in this table, replace the value of SPEC_ID with a value that is not a duplicate of any existing records. You can set the other fields to whatever you want. Theses value will be accessible through the DeviceInfo object.

Example 12-1 PVCDEVSPEC table sample insert

```
insert into PVCDEVSPEC (
   SPEC_ID,
   SPECNAME,
   SESSIONTYPE,
   MAXCONTENTLENGTH,
   MAXURLLENGTH,
   LCDWIDTH,
   LCDHEIGHT,
   LCDCOLORS,
   LCDMONOCHROME,
   IMAGEFORMAT,
   SOUNDFORMAT,
   DOCUMENTFORMAT,
   DOCUMENTVERSION,
   CONTENTDIR,
   DESCRIPTION
) VALUES (
```

```
   100,
   'UP default',
   'UP',
   4096,
   256,
   120,
   80,
   2,
   '1',
   'bmp',
   '',
   'HDML',
   '1.0',
   'hdml_jsp',
   'minimum spec'
);
```

Define relationship between PVCDEVMDL and PVCDEVSPEC

We set up the relationship between the PVCDEVMDL and PVCDEVSPEC tables so that all devices which come from the PvC adapter can share the same setting for content directory (CONTENTDIR) and other information in the PVCDEVSPEC table.

Example 12-2 shows a sample SQL script for inserting a record into the PVCMDLSPEC table in order to define the relationship between the adapter default model and record in the PVCDEVSPEC table.

Example 12-2 PVCMDLSPEC table - sample insert

```
insert into PVCMDLSPEC (
   MDLSPC_ID,
   STOREENT_ID,
   MODEL_ID,
   SPEC_ID
) values (
   100,
   0,
   -1,
   100
);
```

Once these three records are inserted into the corresponding tables, the JSPs will be served from the hdml_jsp directory by the adapter whose name is UP.

12.1.2 Content management configuration

This section provides the procedure for executing the sql script required to update the WCS database tables for content management configuration.

1. Start a DB2 command window by clicking **Start** -> **Programs** -> **IBM DB2** -> **Command window**.
2. Change to the directory of the SQL script. For example, <install_path>\SG246171\direct\hdml\scripts.
3. Back up the original setup_hdml.sql to setup_hdml.sql.org.
4. Modify the setup_hdml.sql script for your environment based on the information provided in the previous section.
5. Execute the setup_hdml.sql script. This script contains insert records for all three required database tables (PVCDEVMDL, PVCDEVSPEC, PVCMDLSPEC):

   ```
   > db2 connect to <wcs_database>
   > db2 -tvf setup_hdml.sql
   ```

   Example 12-3 displays a sample script called setup_hdml.sql, used for content management configuration.

Example 12-3 setup_hdml.sql sample content management configuration script

```
insert into pvcdevmdl (
   MODEL_ID,
   MODELNAME,
   SESSIONTYPE,
   VENDOR,
   DESCRIPTION
) values (
   -1,
   '',
   'UP',
   '',
   ''
);


insert into PVCDEVSPEC (
   SPEC_ID,
   SPECNAME,
   SESSIONTYPE,
   CONTENTDIR
) values (
   100,
   'UP Default',
   'UP',
```

```
   'hdml_jsp'
);


insert into pvcmdlspec (
   MDLSPC_ID,
   STOREENT_ID,
   MODEL_ID,
   SPEC_ID
) values (
   100,
   0,
   -1,
   100
);
```

12.2 Create device-specific content JSPs

This section provides guidance and examples for creating device-specific content JSPs for specific mobile devices. Many of the considerations we discuss are related to design considerations outlined in 10.1, “m-commerce direct application design guidelines” on page 200.

12.2.1 PvC data beans

Many of the JSPs created for the PvC Fashion sample store for m-commerce require that PvC data beans (pvcdatabeans.jar) be deployed. The PvC data beans are not shipped with WCS V5.1 and are not supported. These beans will potentially be made available on the Web at a later date or be included in an upcoming release of WCS V5.x. We have included the pvcdatabeans.jar file in our sample code SG246171.zip.

The pvcdatabeans.jar includes the following beans:

- PVCBufferDataBean

  This data bean allows you to write JSPs that access buffered parameters. If you need to write JSPs with parameter buffering, this data bean is required.

- UserPVCDeviceDataBean

  This data bean allows you to access user device addresses stored in the USERPVCDEV table. If you need to extract data such as e-mail addresses from the USERPVCDEV table, you can use this data bean.

For more details on these data beans, such as available methods, please refer to Appendix C, “PvC data bean reference” on page 355.

12.2.2 Content development

When developing the content-specific JSPs for the targeted mobile device, there are several general development issues to address. Many of these issues have been discussed in 10.1, “m-commerce direct application design guidelines” on page 200.

Example overview

In our example, we started with a working store called PvC Fashion, which supported PC browser clients. The PC browser client JSPs contained HTML content. We decided to keep the navigation of the store similar for PC and mobile clients.

The original device-specific JSPs were developed for i-mode and HDML. Due to confidentiality agreements between IBM and NTT DoCoMo, we are not able to include the i-mode sample. So we created WAP WML content JSPs by converting the HDML to WML. The HDML and WML JSPs are included in the SG246167.zip. The WML JSPs have only been unit-tested through the navigation of category and product display.

Note: All samples included in the SG246167.zip file are offered as is and include no support.

Follow these general content development steps to create device-specific JSPs with HDML content:

1. As a starting point, import all assets into Studio for the PvC Fashion store.

   Refer to the *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167 for details on how to do this.

2. Create a folder in Studio called hdml_jsp under the webapp folder. Next, insert a copy of all the PC browser JSPs into the hdml_jsp directory.

3. Create or modify JSPs to comply with the markup language of the mobile device browser.

   For example, we needed to convert the HTML content in the JSPs to HDML for the target UP.Browser mobile device (UP simulator).

   For the WML content JSPs, we started with the HDML equivalent and used the UP.SDK guidelines on conversion from HDML to WML.

4. Assuming that you have performed the content-management configuration, publish the device-specific JSPs from Studio to the CONTENTDIR on the WCS V5.1 runtime.

At this point, you are ready to test the content developed from the simulator.

The next section explains how we implemented some of the PvC features provided in WCS V5.1 into the content developed for HDML and WML.

12.2.3 Content examples of PvC features

We will describe how to implement the following PvC features available in WebSphere Commerce Suite V5.1:

- Session timeout form (timeout.jsp).
- Password reentry form (relogon.jsp)
- User registration form (UserRegistrationForm.jsp)
- Device registration form (PvcRegisterDevice.jsp)
- Divided large form

Session timeout form (timeout.jsp)

In the PvC Fashion sample (hdml_jsp, wml_jsp), we include timeout.jsp to demonstrate the session timeout form for mobile devices. This form is registered as PVCTimeoutForm in the VIEWREG table.

The timeout function, as related to the timeout form, is available to prevent unauthorized use of mobile devices in the event of loss or theft. The device logging out after a specified period of time protects the individual's information from being stolen when a user is logged on to the server from the mobile device.

Password reentry form (relogon.jsp)

In the PvC Fashion sample (hdml_jsp, wap_jsp) we include the relogon.jsp file, which is registered in the VIEWREG table as ReEnterPasswordForm.

Protection of commands by password reentry is meant to address unauthorized access in the event that the mobile device is lost or stolen. Commands that need protection include those that display pages containing important data such as the user's address and credit card number. This form also protects important processes, such as the order process.

This form is assigned to ReEnterPasswordForm. The ReEnterPasswordForm view is called when protected commands are executed without a password. This view is displayed when a logged-on user executes a protected command without a password. The user is asked to reenter the password to this form. The server will not allow execution of the command without the correct password.

To enable the password lock, register the target command in the XML configuration file.

User registration form (UserRegistrationForm.jsp)

In the PvC Fashion sample (hdml_jsp, wap_jsp), we include the UserRegistrationForm.jsp form, which is registered as UserRegistrationForm in the VIEWREG table. This file internally switches the input form, register.jsp, and form to display for confirming data input by userregister2.jsp.

For a connection via a PvC adapter, you can limit user registration. This is done by configuring the registrationMode for the adapter of the XML configuration file to 1 or 2. This registration form calls the PVCRegistration command that corresponds to the registration limit.

In the new registration form that is offered in our sample, parameters needed by the new registration command RegistrationAdd are passed as arguments to the PVCRegistration command. In addition, the value for the user's e-mail address is passed on as the mobile phone e-mail address of the user. The PVCRegistration command conforms to the given parameter and does both new registration of users and device information registration to the USERPVCDEV and PVCBINDING tables.

Device registration form (PvcRegisterDevice.jsp)

In the PvC Fashion sample (hdml_jsp, wap_jsp) we include the PvcRegistrationDevice.jsp file, which is registered as UseRegistrationForm in the VIEWREG table.

The UserRegistrationForm is necessary when the registrationMode is set to 1 or 2. User that need information entered in this form have registered from adapters other than the PvC adapter, such as from a PC browser client. To make it possible for this kind of user to log in from a mobile device whose connection is managed by the PvC adapter, the relationship between the user ID used on a PC browser client and the mobile device used by that user must be registered.

The e-mail address of a mobile phone can be entered by the user in the device registration form and passed as an argument to the PVCRegisterDevice command. The PVCRegisterDevice command using the parameter saves the user device information in the database. If you configure registrationMode to 1 or

2, this command must be executed the first time a registered user accesses. By using this command, the site manager can collect information such as the e-mail address of a mobile device of a registered user. It is also possible to customize PVCRegisterDevice for the collected data.

This function is effective for mobile devices whose fixed ID can be used, such as a subscriberID. In session control using an unmovable ID, such as a subscriber ID, it is possible to retrieve the user's address even if the user has registered several devices. This is possible by using UserPVCDeviceDatabean.

Divided large form

When a form is too large and its length exceeds the limit of requests that the mobile device can handle, the form must be divided. The PvCFashion sample does not include an example. We have included sample code in this section to illustrate how to address this problem.

Some mobile phone browsers have a limit on the size they can display in one page, or have limits on the length of the requests they can send at one time.

We have prepared the PVCBufferUrl in WCS V5.1 in the event parameters cannot be sent at one time to the target command because of the limit on the length of requests. When using the PVCBufferUrl command, we can divide the sending parameters to the target command. The parameters will be put together at the end and will be sent to the target command to be executed.

Example 12-4, Example 12-5, and Example 12-6 show how PVCBufferURL can be used. This sample divides a data input form into three JSPs. The first JSP (page1.jsp) buffers Parameter_A. The second JSP (page2.jsp) buffers Parameter_B. The third JSP (page3.jsp) buffers Parameter_C, and passes all the buffered parameters to the AddressAdd command at one time.

Example 12-4 PVCBufferUrl sample (page1.jsp)

```
<HTML>
<HEAD>
<TITLE>Page1</TITLE>
</HEAD>
<BODY>
<FORM Name=”PvCBufferUrl" METHOD="POST" Action="PvCBufferUrl">
   <INPUT TYPE="hidden" Name="next" url="page2.jsp"> ..... 1
   <INPUT TYPE="hidden" Name="b_err" VALUE="ErrorPage.jsp"> .....2
   <INPUT TYPE="hidden" Name="b_no" VALUE="b_new,b_no,b_update"> .....3
   <INPUT TYPE="hidden" Name="b_url" VALUE="AddressAdd"> .....4
   <INPUT TYPE="text" Name="Parameter_A" VALUE="">
   <INPUT TYPE="submit" Name="b_new" VALUE="next"> .....5
```

```
</FORM>
</BODY>
</HTML>
```

1 This is the JSP called when the Next button is clicked.
2 This is the JSP called when an error occurs.
3 Specifies the parameters that we do not want to buffer.
4 Specifies the command to execute after buffering.
5 Creates a new buffer.

Example 12-5 PVCBufferUrl sample (page2.jsp)

```
<HTML>
<HEAD>
<TITLE>Page2</TITLE>
</HEAD>
<BODY>
<FORM Name="PvCBufferUrl" METHOD="POST" Action="PvCBufferUrl">
   <INPUT TYPE="hidden" Name="next" url="page3.jsp">
   <INPUT TYPE="hidden" Name="back" url="page1.jsp">
   <INPUT TYPE="hidden" Name="b_err" VALUE="ErrorPage.jsp">
   <INPUT TYPE="hidden" Name="b_no" VALUE="b_new,b_no,b_update">
   <INPUT TYPE="text" Name="Parameter_B" VALUE="">
   <INPUT TYPE="submit" Name="b_update" VALUE="back">
   <INPUT TYPE="submit" Name="b_update" VALUE="next"> .....6
</FORM>
</BODY>
</HTML>
```

6 Adds and renews data on the buffer.

Example 12-6 PVCBufferUrl sample (page3.jsp)

```
<HTML>
<HEAD>
<TITLE>Page3</TITLE>
</HEAD>
<BODY>
<FORM Name="PvCBufferUrl" METHOD="POST" Action="PvCBufferUrl">
   <INPUT TYPE="hidden" Name="back" url="page2.jsp">
   <INPUT TYPE="hidden" Name="b_err" VALUE="ErrorPage.jsp">
   <INPUT TYPE="hidden" Name="b_no" VALUE="b_new,b_no,b_update">
   <INPUT TYPE="text" Name="Parameter_C" VALUE="">
   <INPUT TYPE="submit" Name="b_update" VALUE="back">
   <INPUT TYPE="submit" Name="b_exec" VALUE="next"> .....7
</FORM>
</BODY>
</HTML>
```

7 All buffered parameters specified in 4 above are passed to the command and executed.

12.3 Deploy content

There are several methods of deploying the JSPs. For the purposes of development, we published the JSPs directly from WebSphere Studio, as documented in *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167.

Once the content is deployed, you can begin testing from mobile device simulators, PC browser clients, and eventually with real mobile device hardware.

12.4 Advanced content and configuration example

This section provides an example of more advanced content creation and content management configuration. Using these examples, we describe how automatic content selection works.

12.4.1 Auto selection of content

For example, suppose there are the following necessary registration settings for auto selection in the database:

- Only devices that have the model name T250 are mapped to the specification that has reference number 11.
- Other devices that connect via the UP PvC adapter are mapped to the specification that has the reference number 10.

You can get a simulator configuration file for a device named T250 from the following URL:

```
http://developer.phone.com/download/simconfig.html
```

By changing the configuration for your simulator, you can simulate access from a device that has the model name T250.

Table 12-1 PVCDEVMDL sample values

MODEL_ID	SESSIONTYPE	MODELNAME	other fields
-1	UPG		Any
-2	UPG	T250	Any

Table 12-2 PVCDEVSPEC sample values

SPEC_ID	SESSIONTYPE	CONTENTDIR	other fields
10	UPG	hdml	Any
11	UPG	hdml_high_res	Any

Table 12-3 PVCMDLSPEC

MDLSPEC_ID	MODEL_ID	SPEC_ID	STOREENT_ID
100	-1	10	0
101	-2	11	0

Then, assume that there was the following entry for the LogonForm view.

Table 12-4 VIEWREG

VIEWNAME	DEVICEFMT_ID	PROPERTIES	STOREENT_ID
LogonForm	-1	LogonForm.jsp	0

If the settings in the above are inserted, the results of access to the URL are:

```
http://hostname/webapp/wcs/stores/servlet/LogonForm?storeId=10001
```

which contains the store ID parameter, as seen in Table 12-5.

Table 12-5 Results of access

Model	Used adapter	JSP loaded
T250	UPG	PvCFashion/hdml_high_res/LogonForm.jsp
Other UPG devices	UPG	PvCFashion/hdml/LogonForm.jsp
Netscape of IE on PC		PvCFashion/LogonForm.jsp

Results of an access to a URL such as:

```
http://hostname/webapp/wcs/stores/servlet/LogonForm
```

or

```
http://hostname/webapp/wcs/stores/servlet/LogonForm?storeId=10001&storeDir=no
```

which are not set up to append a store directory in front of a JSP search path, will become as seen in Table 12-6.

Table 12-6 Results with no append

Model	Used adapter	JSP loaded
T250	UPG	hdml_high_res/LogonForm.jsp
Other UPG devices	UPG	hdml/LogonForm.jsp
Netscape of IE on PC		LogonForm.jsp

Thus, for access from a PC browser such as Netscape Navigator or Microsoft Internet Explorer, the server acts as usual. For mobile phones, the server adds the value of CONTENTDIR of the related record in the PVCDEVSPEC table to the JSP file search path.

12.4.2 Reflection of terminal specification to JSP view

The WCS server allows another method of content management. This is an approach from the JSP source code. For devices that connect via a PVC adapter, information stored in the PVCDEVSPEC table corresponding to the model of the device are accessible through an object. This information is set as an instance object of com.ibm.commerce.DeviceInfo class as a request attribute of the JSP file to be processed.

Using the DeviceInfo class object, you can access the columns as seen in Table 12-7.

Table 12-7 DeviceInfo class object access

Type	Name	Corresponding database field	Description
String	pvcSessionId	PVCSESSION.PVCSESSION_ID	Reference number of pervasive session information in the PVCSESSION table
String	pvcSessionType	PVCSESSION.SESSIONTYPE	Name of the adapter that handled this session
String	model	PVCDEVMDL.MODELNAME	Model name of the client's device
String	vendor	PVCDEVMDL.VENDOR	Vendor name of the model

Type	Name	Corresponding database field	Description
int	maxContentLength	PVCDEVSPEC.MAXCONTENTLENGTH	Maximum length of the content the device can receive
int	maxUrlLength	PVCDEVSPEC.MAXURLLENGTH	Maximum length of the URL the device can request
int	width	PVCDEVSPEC.LCDWIDTH	Width of the LCD panel
int	height	PVCDEVSPEC.LCDHEIGHT	Height of the LCD panel
int	colors	PVCDEVSPEC.LCDCOLORS	Number of displayable colors of the LCD panel
boolean	isMonochrome	PVCDEVSPEC.LCDMONOCHROME	Indicates whether LCD panel is monochrome
String	imageFormat	PVCDEVSPEC.IMAGEFORMAT	Supported format for images
String	soundFormat	PVCDEVSPEC.SOUNDFORMAT	Supported format for sounds
String	documentFormat	PVCDEVSPEC.DOCUMENTFORMAT	Supported format for documents
String	documentVersion	PVCDEVSPEC.DOCUMENTVERSION	Supported version of document format
String	contentDirectory	PVCDEVSPEC.CONTENTDIR	Name of the directory where contents for the model are located
String	specDescription	PVCDEVSPEC.DESCRIPTION	Description of the specification
String	modelDescription	PVCDEVMDL.DESCRIPTION	Description of the model

The code in Example 12-7 shows how to use the DeviceInfo object information in a JSP. Using the jsp:useBean tag, you can get the DeviceInfo object that stores information about the client's device. If your database is configured properly, a designer can write codes to display the best result for each browser of a different specification using this information from the DeviceInfo object.

Example 12-7 Sample code to use DeviceInfo object

```
<!--@Example: How to change GIF image depends on terminal capability>
<jsp:useBean id="pvc_device_info" scope="request"
class="com.ibm.commerce.pvcadapter.DeviceInfo">
</jsp:useBean>
<HTML>
<BODY>
<% if ( pvc_device_info.isMonochrome ) { %>
   <IMG SRC="mono.gif">
   This page is designed suitable for monochrome terminal.
<% } else { %>
   <IMG SRC="color.gif">
   This page is designed suitable for color terminal.
<% } %>
</BODY>
</HTML>
```

12.4.3 Content managed by adapter

If you are planning to prepare a PvC adapter, you need to do the same configuration described in 12.1, “Content management configuration” on page 226, for each adapter with the specific adapter definitions.

12.4.4 Content managed by model

In this section, we explain how to prepare different JSPs or different specifications for the DeviceInfo bean for each categorized group of device models. We discuss how to categorize similar specification of models and how to manage contents using this categorization. This explanation is given in 12.1, “Content management configuration” on page 226.

Prerequisites

This section includes the prerequisites needed prior to content management configuration and content development, as follows:

- You need at least one store installed.

  We recommend that you install our sample store (PvC Fashion) on the WCS server.

- UP.SDK simulator should be installed and configured.

If you installed the UP.SDK simulator, you may have a directory called configs in the simulator directory. You can switch the configuration for your simulator by selecting **File** -> **Open Configurations...** from the menu. In this directory, there are several configuration files and you will find a file named generic.pho. In this file you will see the following line:

```
DEVICEID UPG1
```

This setting corresponds to the name of the device model that is simulated. If you are using this configuration, model name of your device should be UPG1. Next copy these files as IBM1.pho, IBM2.pho, IBM3.pho and IBM4.pho. Then modify the DEVICEID field, as seen in Table 12-8.

Using these settings, which have different model names, we can verify our PvC adapter detection and content management configuration.

Table 12-8 DEVICEID field value

IBM1.pho	IBM01
IBM2.pho	IBM02
IBM3.pho	IBM03
IBM4.pho	IBM04

Configuration example

We have categorized the device models into 3 groups, and need to configure our site to map these groups to different specifications, as seen in Table 12-9.

Table 12-9 Group specifications

Group	Minimum Specification
UPG_Color1	Color and high resolution (Minimum spec is 180x140 pixel, 256 colors)
UPG_Color2	Color and normal resolution (Minimum spec is 100x80 pixel, 256 colors)
UPG_Default	Other all devices (minimum spec is 100x80 pixel, 2 monochrome colors)

For these groups we prepare the following settings for content management:

- UPG_Color1 has different resolution from other groups, so we prepare a separate JSP for this group.
- In resolution, there's no difference in UPG_Color2 and UPG_Default, so we prepare a common JSP for these groups and make color and monochrome changes by using the DeviceInfo bean.

- We apply these setting only for a store with a 10001 store ID.

Table 12-10 and Table 12-11 provide a summary of our configuration.

Table 12-10 Sample categorization for content management

Simulator config file	Model name adapter returns	Group
IBM01.pho	IBM01	UPG_Color1
IBM02.pho	IBM02	UPG_Color2
Other devices from UPG adapter	depends on device	UPG_Default

Table 12-11 Content directory and minimum spec for each group

Group	Content Directory	Minimum Specification
UPG_Color1	UPG_Color1	180x140 pixel 256 colors
UPG_Color2	UPG_Default	100x80 pixel 256 colors
UPG_Default	UPG_Default	100x80 pixel monochrome

The relationship between model and specification includes the content directory. If we define a relationship model and specification for our sample configuration into the PVCDEVMDL, PVCDEVSPEC and PVCMDLSPEC tables, it should look like Figure 12-1.

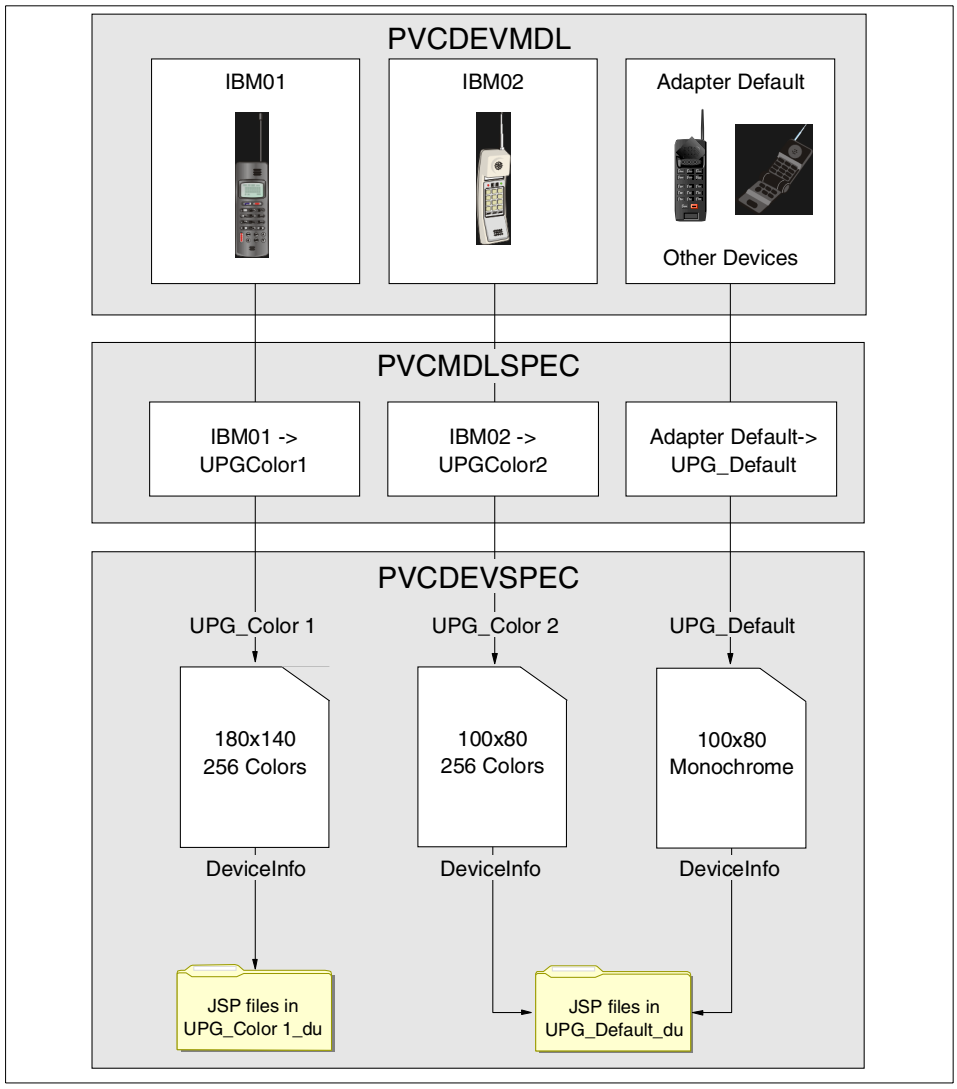


Figure 12-1 PvC tables configuration

PVCDEVMDL database table

First we need to register the device model that needs to be categorized. We need to register the models that have a specific model-spec mapping adapter default model record that represents all other models that do not have specific mapping, as seen in Table 12-12.

Table 12-12 PVCDEVMDL settings

PVCDEVMDLMODEL_ID	SESSIONTYPE	MODELNAME
-10	UPG	
-11	UPG	IBM01
-12	UPG	IBM02

SQL statements for these settings are seen in Example 12-8.

Example 12-8 Sample SQL insert records for PVCDEVML table

```
insert into PVCDEVMDL ( MODEL_ID,SESSIONTYPE,MODELNAME) values (-10, 'UPG',
'');
insert into PVCDEVMDL ( MODEL_ID,SESSIONTYPE,MODELNAME) values (-11, 'UPG',
'IBM01');
insert into PVCDEVMDL ( MODEL_ID,SESSIONTYPE,MODELNAME) values (-12, 'UPG',
'IBM02');
```

Note: Before running the SQL statements, we recommend that you remove the old records in the PVCDEVMDL, PVCDEVSPEC and PVCMDLSPEC tables to avoid errors when you run these statements.

PVCDEVSPEC database table

Then we need to register the content directory and minimum specifications for each group. We need to insert information into the PVCDEVSPEC table, as seen in Table 12-13.

Table 12-13 PVCDEVSPEC settings

SPEC_ID	SPECNAME	SESSION_TYPE	CONTENTDIR	LCDWIDTH/ LCDHEIGHT/ LCDCOLORS/ LCDMONOCHROME
100	UPG_Color1	UPG	UPG_Color1_dir	180/100/256/0

SPEC_ID	SPECNAME	SESSION_TYPE	CONTENTDIR	LCDWIDTH/ LCDHEIGHT/ LCDCOLORS/ LCDMONOCHROME
101	UPG_Color2	UPG	UPG_Default_dir	100/80/256/0
102	UPG_Default	UPG	UPG_Default_dir	100/80/256/1

The SQL statements for these settings are seen in Example 12-9.

Example 12-9 Example SQL statements to insert records in to PVCDEVSPEC

```
insert into PVCDEVSPEC (SPEC_ID, SPECNAME, SESSIONTYPE, MAXCONTENTLENGTH,
MAXURLLENGTH, CONTENTDIR, LCDWIDTH, LCDHEIGHT, LCDCOLORS, LCDMONOCHROME)
  values ( 100, 'UPG_Color1', 'UPG', 5242880,255, 'UPG_Color1_dir' , 180, 100,
256 , '0');
insert into PVCDEVSPEC (SPEC_ID, SPECNAME, SESSIONTYPE, MAXCONTENTLENGTH,
MAXURLLENGTH, CONTENTDIR, LCDWIDTH, LCDHEIGHT, LCDCOLORS, LCDMONOCHROME)
  values ( 101, 'UPG_Color2', 'UPG', 5242880,255, 'UPG_Default_dir' , 100, 80,
256 , '0');
insert into PVCDEVSPEC (SPEC_ID, SPECNAME, SESSIONTYPE, MAXCONTENTLENGTH,
MAXURLLENGTH, CONTENTDIR, LCDWIDTH, LCDHEIGHT, LCDCOLORS, LCDMONOCHROME)
  values ( 102, 'UPG_Default','UPG', 5242880,255, 'UPG_Default_dir' , 100, 80,
2 , '1');
```

PVCMDLSPEC database table

For each group you need to define the relationship between the records in the PVCDEVMDL and the PVCDEVSPEC tables. In our sample, we need to enable this setting only for the store that has a store ID of 10001 as seen in Table 12-14.

Table 12-14 Example settings for PVCMDLSPEC

MDLSPEC_ID	MODEL_ID	SPEC_ID	STOREENT_ID
1000	-10	102	10001
1001	-11	100	10001
1002	-12	101	10001

The SQL statements for these settings are seen in Example 12-10.

Example 12-10 Setup model-spec mapping

```
insert into PVCMDLSPEC(MDLSPC_ID,MODEL_ID,SPEC_ID,STOREENT_ID) values (1000,
-10, 102, 10001 );
insert into PVCMDLSPEC(MDLSPC_ID,MODEL_ID,SPEC_ID,STOREENT_ID) values (1001,
-11, 100, 10001 );
insert into PVCMDLSPEC(MDLSPC_ID,MODEL_ID,SPEC_ID,STOREENT_ID) values (1002,
-12, 101, 10001 );
```

Testing the configuration

Now you can verify that these configurations are in effect by accessing the view from different model names of devices. For example, insert one record into the VIEWREG table, as seen in Example 12-11.

Example 12-11 Setting up the view registry

```
insert into viewreg (
   VIEWNAME,
   DEVICEFMT_ID,
   STOREENT_ID,
   INTERFACENAME,
   CLASSNAME,
   PROPERTIES,
   HTTPS,
   INTERNAL
) values (
   'Test',
   -1,
   10001,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=test.jsp',
   0,
   0
);
```

We also prepared three JSPs called test.jsp in two separate directories, and one in the document root. We assume that the store directory for the store that has store ID 10001 is PvCFashion. If you have already installed our sample, you might have this directory in your document root.

For PC browsers we prepared a test.jsp under the store root directory.

Example 12-12 Page for PC browser. (PvCFashion/test.jsp)

```
<%@ page language="JAVA"%>

<HTML>
   /PvCFashion/test.jsp <BR>
   Page for PC browser.
</HTML>
```

For the UPG_Color1 group, we prepared a test.jsp in the UPG_Color1_dir directory, which is under the store directory, as follows:

Example 12-13 test.jsp for UPG_Color1

```
<%@ page import="com.ibm.commerce.pvcadapter.*"  %>
<%@ page contentType="text/vnd.wap.wml; charset=iso-8859-1" %>

<jsp:useBean id="pvc_device_info" scope="request"
class="com.ibm.commerce.pvcadapter.DeviceInfo">
</jsp:useBean>

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD">
<wml>
<card>
<p>
Page for IBM01 from UPLink Gateway.<br/>
Your terminal must have at least following capability.  <br/>

 Model: <%= pvc_device_info.model %><br/>
 Width:  <%= pvc_device_info.width %> <br/>
 Height: <%= pvc_device_info.height %><br/>
 Colors: <%= pvc_device_info.colors %><br/>

We'll provide you color contents.<br/>
Is color:  Yes <br/>

</p>
</card>
</wml>
```

For the UPG_Color2 and UPG_Default devices we prepared a test.jsp in the UPG_Default_dir, which is under the store directory.

Example 12-14 test.jsp for UPG_Default_dir

```
<%@ page import="com.ibm.commerce.pvcadapter.*"   %>
<%@ page contentType="text/vnd.wap.wml; charset=iso-8859-1" %>
```

```
<jsp:useBean id="pvc_device_info" scope="request"
class="com.ibm.commerce.pvcadapter.DeviceInfo">
</jsp:useBean>

<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD">
<wml>
<card>
<p>
Default page for devices from UPLink Gateway.<br/>
Your terminal must have at least following capability.  <br/>

 Model: <%= pvc_device_info.model %><br/>
 Width:  <%= pvc_device_info.width %> <br/>
 Height: <%= pvc_device_info.height %><br/>
 Colors: <%= pvc_device_info.colors %><br/>

 <% if (!pvc_device_info.isMonochrome) { %>
We'll provide you color contents.<br/>
Is color:  Yes <br/>

 <% } else { %>
We'll provide you monochrome contents.
 Is color:  No<br/>
 <% } %>
</p>
</card>
</wml>
```

Now we have three different JSPs for the same view registry in the VIEWREG table. Now let's try to access the following URL from different devices to see if the WCS server selects the appropriate content for each browser:

```
http://localhost/webapp/wcs/stores/servlet/Test?storeId=10001
```

PC browser test

If you try to access from a PC browser client, the WCS server should process the test.jsp in the root PvCFashion directory. The result should look like Figure 12-2:

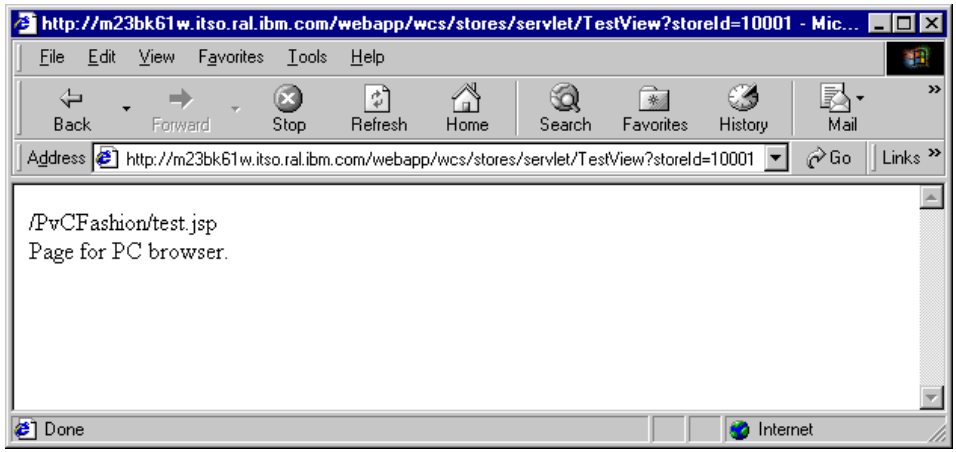


Figure 12-2 PC browser results page of test.jsp

IBM01 simulator test

Next, try to access the store from the IBM01 device. To emulate this model of the device, you need to switch the simulator configuration to IBM01.pho. If you try to access from the IBM01 device, the WCS server should process the test.jsp in the UPG_Color1 directory. The result should be displayed in the simulator, as seen in Figure 12-3.

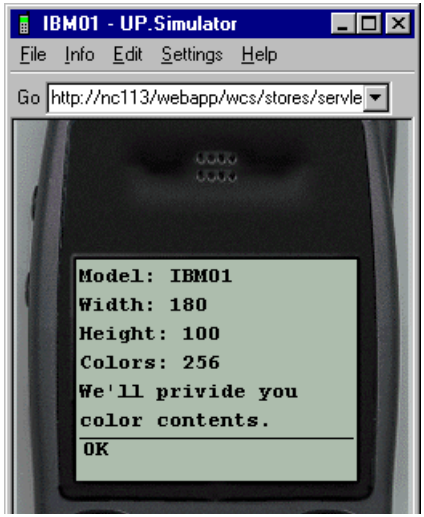


Figure 12-3 IBM01 simulator result

IBM02 simulator test

Repeat the test used in IBM01 for IBM02 (configure for IBM02). The result should look like Figure 12-4.

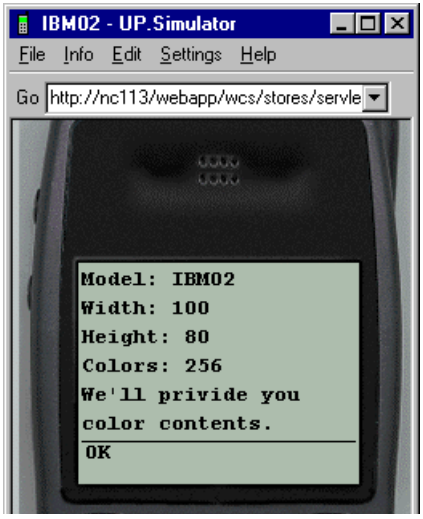


Figure 12-4 IBM02 simulator result

IBM03 and IBM04 simulator test

If you try other UPG devices, for example if you choose the setting for IBM03 or IBM04, the server will pick up the JSP in the PvCFashion/UPG_Default_dir directory according to the mode-spec mapping shown in Example 12-10. The result should look like Figure 12-5.

You can make sure that the information in the PVCDEVSPEC table for the 'UPGDefault' devices is passed to the JSP by checking the message `We'll provide you monochrome contents` displayed on the simulator.

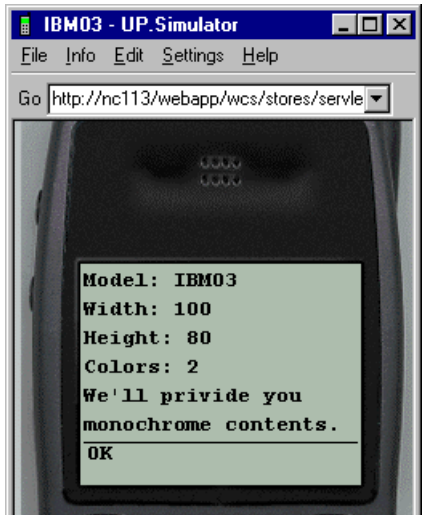


Figure 12-5 IBM03 simulator result

13

Creating custom PvC commands

WebSphere Commerce Suite V5.1 includes several PvC specific commands for merchants who wish to manage mobile user and mobile device information. This chapter describes how to create your own custom PvC commands using the PvC commands PVCRegistration, PVCRegisterDevice, PVCChangeDevice supplied with WebSphere Commerce Suite V5.1.

The chapter is organized into the following topics:

- WCS V5.1 PvC command overview
- Create and deploy a custom PvC command

13.1 WCS V5.1 PvC command overview

This section provides a brief description of the PvC commands included in WebSphere Commerce Suite V5.1, and describes how they are used.

For more detailed reference information on the PvC commands, refer to Appendix B, “PvC command reference” on page 339.

13.1.1 PvC command summary

This section includes a brief description of the PvC commands.

- PVCRegistration

  This command is used to enable registration and renewal of the PvC device information of users. Registration records of users and PvC device information records can be entered and updated using this command. This command is used together with Secure Socket Layer (SSL) to encode the user's logonID, password and individual information.

  The PVCRegistration command is required when the PvC adapter definition registrationMode is set higher than 1.

- PVCRegistrationDevice

  This command enables the registration and update of PvC information for users that have already registered (for example, from a PC browser client). This command is used together with SSL to encode a user's logon ID, password and individual information.

  The PVCRegistrationDevice command is required when the PvC adapter definition registrationMode is set higher than 1.

- PVCChangeDevice

  If your site is configured to manage PvC devices as one device per user because the session registrationMode is set to 2 in the XML configuration file, you need to manage canceled or changed devices. This is necessary, because once a user registers his/her device, no further registration is allowed using another device. The account in the WCS database is locked until the old user-device relationship is canceled.

  This command is required when the PvC adapter definition registrationMode equals 2.

- PVCBufferUrl

  This command enables buffering of input field data placed in multiple pages, and then sends the data to one command. It temporarily saves parameters of the destination URL in the WCS database PVCBUFFER table. Buffers that have not been updated for some time will be made unavailable by specifying a bufferTimeout (in minutes) in the configuration file.

- ReEnterPassword

  This command adds the given reenterpw parameter to the specified URL and redirects it. Usually the ReEnterPassword command is used by a JSP which is assigned to ReEnterPasswordForm. ReEnterPasswordForm is called when executing the command without a password in locked password status.

13.1.2 Scenarios for using the PvC commands

If you have already developed a store for PC browser clients, and wish to enable your store for mobile devices, you may wish to implement the PvC commands provided in WCS V5.1.

Imagine that a company called PvC Fashion has a B2C store currently available for PC browser clients and now want to enable the store for m-commerce. The IT staff assumes that many of the customers using mobile devices to access the store have already registered and have an active account in the WCS database. They want to allow the user to add such information as the e-mail address of their mobile phone; they also want to provide mobile users constant access to the store.

To address this business requirement for users originally registered via a PC browser client, WCS V5.1 has included a PvC command called PVCRegistrationDevice. By setting the registrationMode for the PvC adapter definition to 1 or 2, we can require the user to enter additional registration information by using the PVCRegisterDevice command. Once this information is entered, we can provide services for the mobile user.

For new customers who only use mobile phones, you can use the PVCRegistration command to allow your customers to login to your WCS server. The PVCRegistration executes the PVCRegisterDevice command internally after normal registration.

In both cases, the PVCRegistrationDevice and PVCRegistration commands create records in the USERPVCDEV and PVCBINDING tables. If you want to use the address information of your customer, you can check the USERPCDEV table. The PVCBINDING table is used by the system when a logon is executed by WCS.

After the logon is processed, the server checks if the mobile device attempting to log on has its user listed in the PVCBINDING table. If the user does not have a record in the PVCBINDING table, this means that the user has not finished the device registration. In this case, the server rejects the logon, even if the user has entered and sent the correct user ID and password.

PVCChangeDevice is a more specialized command. You can use this command only when registrationMode for the adapter is set to 2. In this case, the relationship between shoppers and devices must be strictly one-to-one. In other words, once a user registers his/her device, he/she cannot use any other device for logon. In addition, the device registered cannot be used by any other user. This can be very restrictive, and should be used with discretion.

For example, let us say that we only want to require the user to enter the logon ID for registration. In the future we do not want to ask the customer to enter the user ID on the logon form, but only to enter the password for logon. Once the user ID is entered, the user is notified by e-mail when the registration is complete.

To enable this command, we need to define the one-to-one relationship between the WCS user and mobile phone subscriber ID. Unfortunately, the subscriber ID can in some cases be changed by the customer. In other cases, the subscriber ID is provided by the carrier with the new phone and cannot be changed.

To succeed in registering information for a new device, we can use the PVCChangeDevice command to break the old relationship between a user and a mobile phone, and to create a new record. This command also receives addresses for the device and stores them in the USERPVCDEV table.

By default, there is no special checking for the PVCRegisterDevice and PVCChangeDevice commands. If you want a condition to allow a user to log in to your WCS server, you can customize it by extending an existing command. In the following section, by creating our own custom PvC command, we will demonstrate how to extend the PVCRegisterDevice command to enable a custom parameter checking for device registration.

13.2 Create and deploy a custom PvC command

In this section, we demonstrate how to create and deploy a custom PvC command by extending existing PvC commands.

Sample code for custom PvC command

The sample JAR code (included class and Java) for the following custom PvC command can be found in the <install_path>\SG246171\command.pvc directory after unzipping the SG246171.zip file.

13.2.1 Create a custom PvC command

For this example, let us assume that we have the following application code:

- A command to accept user e-mail address input.

  This command has the function of checking the validity of an e-mail address and also of keeping a list of valid e-mail addresses. Checking the validity of e-mail addresses is done by sending back e-mail which has a link to the command which registers submitted e-mail addresses. All users need to finish this check before registration.

- An access bean and EJB to access the list of e-mail addresses.

  The access bean ValidEMailAddressAccessBean, and EJB are used to access the list of valid e-mail addresses. The ValidEMailAddressAccessBean uses the method findByAddress(String) to search valid addresses from the list.

Next, we describe how to write the code which satisfies the following conditions:

- The type of address is 'E'
- The e-mail address in the parameter is found in the list.

To create the custom PvC command, complete the following steps:

1. Start VAJ.
2. Create a project.

   For example, we created a project named `Sample customize parameter check`.

3. Create a package.

   For example, we created a package named `com.ibm.commerce.sample.pvc.command`.

4. Create a class.

   For example, we created a class named `PVCRegisterDeviceCmdmpl`.

Next, we need to customize this method to do our custom parameter checks. The super class already has a method which performs some simple parameter checks. This method is defined by the super class as follows:

```
public void checkParameters()
```

You can see that the super class has this method already by opening the class browser. We need to override the method. In this method, we write logic to check if parameters are valid.

To add this method to the subclass, follow these next steps:

5. Right-click **MyPVCRegisterDeviceCmdImpl**.
6. Select **Add** -> **Method**.
7. When the Add Method window appears, enter `public void checkParameters().`
8. Click **Next** twice.
9. Add `com.ibm.commerce.exception.ECException.`
10. Click **Finish**.

You now have a method check parameter for MyPVCRegisterDeviceCmdImpl. You can write your own custom parameter checks.

This class has static members for error codes and a dummy class, which is a substitution of the actual access bean and checkParameter method used for customization.

In the beginning of checkParameter, parameters are checked using methods prepared by the super class. Refer to “PVCRegistrationDevice” on page 344 for details on the super class.

After checking parameters with the checkParameter method, which is provided by the super class, this command checks the mandatory parameter ‘devaddress1’, which is present in the request. If the value of ‘devaddress1’ is not found in the request, this command throws an exception with the following error code: ERR_CODE_MISSING_ADDRESS1.

Then the command checks if the address type is ‘E’ or not. If the address type is not the same as expected, this command throws an exception with the error code ERR_CODE_BAD_ADDRTYPE1.

After completing the checks above, the command checks if the value of devaddress1 is listed in the database as a valid e-mail address. Instead of writing an EJB and access bean code, we prepared a dummy class which has only the method we are expecting from the access bean to simplify the explanation of steps on how to extends parameter checking.

The sample MyPVCRegisterDeviceCmdImpl code can be seen in Example 13-2.

Example 13-1 MyPVCRegisterDeviceCmdImpl

```
package com.ibm.commerce.sample.pvc.command;

import javax.ejb.*;
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.ras.*;
import com.ibm.commerce.server.*;
```

```
import com.ibm.commerce.exception.*;

public class MyPVCRegisterDeviceCmdImpl extends
com.ibm.commerce.pvc.commands.PVCRegisterDeviceCmdImpl {

   public static final String ERR_CODE_BAD_ADDRTYPE1 =
"ERR_CODE_BAD_ADDRTYPE1";
   public static final String ERR_CODE_BAD_ADDRESS1 = "ERR_CODE_BAD_ADDRESS1";
   public static final String ERR_CODE_MISSING_ADDRESS1 =
"ERR_CODE_MISSING_ADDRESS1";
   public static final String ERR_CODE_NOT_IN_LIST = "ERR_CODE_NOT_IN_LIST";

   // Dummy class for EMailCheckAccessBean
   class ValidEMailAddressAccessBean  {
      ValidEMailAddressAccessBean findByAddress(String s) throws
ObjectNotFoundException {
         if (true) throw new ObjectNotFoundException();
         return null;
      }
   }

   public MyPVCRegisterDeviceCmdImpl() {
      super();
   }

   public void checkParameters() throws com.ibm.commerce.exception.ECException
{
      String className = this.getClass().getName();
      String methodName = "checkParameters";

      super.checkParameters();

      if (getAddress1() == null) {
         TypedProperty hshNVPs = new TypedProperty();
         hshNVPs.put( ECConstants.EC_ERROR_CODE, ERR_CODE_MISSING_ADDRESS1);
         throw new ECApplicationException(
                           ECMessage._ERR_MISSING_CMD_PARAMETER,
                           className,
                           methodName,
                           ECMessageHelper.generateMsgParms("devaddress1"),
                           ERRTASK_NAME,
                           hshNVPs);
      }
      if (!getAddrType1().equals("E")) {
         TypedProperty hshNVPs = new TypedProperty();
         hshNVPs.put( ECConstants.EC_ERROR_CODE, ERR_CODE_BAD_ADDRTYPE1);
         throw new ECApplicationException(
                           ECMessage._ERR_BAD_PARMS,
                           className,
```

```
                            methodName,
                            ECMessageHelper.generateMsgParms("devaddrtype1"),
                            ERRTASK_NAME,
                            hshNVPs);
        }

        ValidEMailAddressAccessBean addr= new ValidEMailAddressAccessBean();
        try {
           addr = addr.findByAddress(getAddress1());
        } catch (ObjectNotFoundException e) {
           TypedProperty hshNVPs = new TypedProperty();
           hshNVPs.put( ECConstants.EC_ERROR_CODE, ERR_CODE_NOT_IN_LIST);
           new ECApplicationException(
                            ECMessage._ERR_BAD_PARMS,
                            className,
                            methodName,
                            ECMessageHelper.generateMsgParms("devaddress1"),
                            ERRTASK_NAME,
                            hshNVPs);
        }
     }
}
```

13.2.2 Deploy the custom PvC command

In order to deploy a PvC command, the following steps must be completed:

- Export the command to a JAR
- Deploy the JAR file
- Update the application server classpath
- Update the WCS database CMDREG table

Export the command to a JAR

After developing the custom PvC command code, we need to export the code to a JAR file.

For example, we exported the package `com.ibm.commerce.sample.pvc.command` to a JAR file named `MyPVCRegisterDeviceCmdImpl`.jar.

Deploy the JAR file

Copy the JAR file to the class path of the <wcs_install_path>\lib directory.

For example, we copied the `MyPVCRegisterDeviceCmdImpl`.jar to the `c:\ibm\wcs\lib` directory.

Update the application server classpath

Next, we need to update the WebSphere Commerce Server - <instance> command line arguments (classpath) field.

For example, in the WebSphere Administrator's Console, we added `c:\ibm\wcs\lib\MyPVCRegisterDeviceCmdImpl.jar` to the command line arguments field for the WebSphere Commerce Server - <instance>.

Once the classpath (command line arguments) has been updated, we need to stop, then restart the WebSphere Commerce Server - <instance>.

Update the WCS database CMDREG table

The WCS database CMDREG table needs to be updated so that the server can execute MyPVCRegisterCmdImpl instead of PVCRegisterCmdImpl, which is already installed on your system.

To change the database registration for the new PvC command, run the SQL script as seen in Example 13-2.

Example 13-2 cmdreg update script

```
update cmdreg
set CLASSNAME='com.ibm.commerce.sample.pvc.command.MyPVCRegisterDeviceCmdImpl'
where INTERFACENAME='com.ibm.commerce.pvc.commands.PVCRegisterDeviceCmd';
```

This SQL statement updates the entry for the command whose interface name is `com.ibm.commrce.pvc.commands.PVCRegisterDeviceCmd`. After running this script, you need to stop, then restart the WebSphere Commerce Server - <instance> to enable the MyPVCRegisterDeviceCmdImpl command when the PVCRegistration or PVCRegisterDevice URL command is executed.

14

HDML implementation sample

This chapter provides guidelines specific to supporting HDML browser-based mobile clients using the m-commerce direct approach.

This chapter is organized into the following topics:

- HDML toolkits and test clients
- Sample code for HDML
- m-commerce direct development for HDML

14.1 HDML toolkits and test clients

This section provide information specific to developing mobile applications that use HDML browsers. We show development kits that provide a simulator with HDML browsers used for testing. In addition, we show real wireless devices that support HDML browsers used for testing.

14.1.1 UP.SDK V3.2 for HDML

OpenWave Phone.com, formally Unwired Planet, provides a Software Developer's Kit (SDK) that includes a simulator that supports HDML 3.0 content through its microbrowser, examples, and supporting documentation.

Although Nokia is dominant in Europe, the UP Phone.com browser is used by many phone manufacturers. It appears that HDML-based sites in the US and Japan are shifting to WAP/WML or imode/CHTML,respectively. HDML has a large install base and should not be ignored if building m-commerce Web sites for the current mobile market.

There are currently two versions of the UP.SDK that support HDML content:

- UP.SDK V3.2 for HDML
- UP.SDK V4.1 via UP.Link translation service

Note: We found that the UP.SDK V3.2 for HDML for Windows provided better support for our test needs.

Downloading UP.SDK V3.2 for HDML

You can download the UP.SDK V3.2 from the following URL:

`http://developer.phone.com/download/license_32.html`

Contents of UP.SDK V3.2 for HDML

The Windows-based UP.SDK V3.2 for HDML contains the following:

- Up.Simulator and debugging console
- Example code
- Documentation
  - UP.SDK Getting Started Guide (PDF)
  - UP.SDK Developer's Guide (PDF)
  - UP.SDK Tools & APIs Reference (PDF)

– UP.SDK HDML 3.0 Language Reference (PDF)

Installing UP.SDK V3.2 for HDML

To install the UP.SDK, follow these steps:

1. Click the download executable file called upsdkw32he.exe
2. Follow the installation instructions provided.
3. After you install the UP.SDK, click **Start** -> **Programs** -> **UP.SDK 3.2 for HDML** -> **Up.Simulator**. The simulator should be displayed as seen in Figure 14-1.

Figure 14-1 UP.Simulator V3.2 for HDML

Configuring UP.SDK V3.2 for HDML

After installing the UP.SDK by clicking the executable download file, you should configure the simulator for your test needs. The UP.Simulator can be configured for different modes of operation and emulation modes of different mobile phones.

Modes of operation

- HTTP Direct

  Configured to connect directly to the application (no gateway)

- UP.Link

  This type of configuration is valid if your application requires a unique subscriber ID. For example, the service provider EZ in Japan provides a subscriber ID via the provider gateway. When creating the application and testing it, you will need to configure the simulator to use the UP.Link.

  It is quite important, in creating the adapter, to know how the subscriber ID is included in the request from the browser. Since the simulator mimics access from UP.Link Gateway, examining this gateway's specifications allows us to see how the subscriber ID is used.

  For more information about UP.Link Gateway's functions and contents, please refer to the *UP.SDK Developer's Guide* found at the following URL:

  `http://developer.phone.com/doc/31w/devgd.pdf`

Mobile phone emulation of the configuration file

As mentioned, many phone manufactures use the UP.Browser. In order to provide emulation for each manufacturer, the UP.Simulator can load phone configuration files. The phone configuration file contains information about the size of the screen, as well as input to emulate the manufacturer's device.

Subscriber ID

According to the UP.SDK Developer's Guide, a unique device ID value is set x-up-subno. This is found before the blank of the second section, where the browser version is divided by '/'. Since the UP.Simulator is not an actual phone but a simulator, the value of x-up-subno is a combination of the username and the host name of the machine that executes the simulator. These will be connected with '_'.

Also, the value of User-Agent varies according to the configuration file of the simulator. Since various configuration files for mobile phones are provided at the download site of the toolkit (simulator), the model name will vary according to the different kinds of mobile phone emulators used by the simulator. This is possible by replacing the configuration file of the simulator to use a different mobile phone emulator. In this chapter, we will explain how to make a concrete adapter by making a new adapter named UPLinkGWAdapter, which extracts the value of x-up-subno as a subscriber ID and part of User-Agent as a model name.

Verifying UP.Simulator

It is important that you verify that the UP.Simulator is installed and configured correctly prior to using it for development. A simple test provides a basic verification that the simulator is working properly.

Useful information from OpenWave

OpenWave provides a wealth of information about mobile computing, which can be found at: `http://developer.phone.com/`

We have included a URL that contains information about markup languages:

`http://developer.phone.com/resources/markup.html/`

Note: The UP.SDK V3.2 for WML, contains information for converting content from HDML to WML that we found useful. Beware that some of the extensions provided by the UP.Browser are not supported by standard WAP browsers, specifically not part of the formally WAP specification.

14.1.2 Mobile device hardware

As mentioned in the previous section, many mobile phone manufacturers use the UP.Browser, which can support HDML and WML. You can visit various phone manufacturers' Web sites to find a phone that is right for you.

We used a Motorola V.Series V2282 mobile phone in our real wireless hardware intranet test environment. We found this to be a good match for our real hardware testing. This particular phone uses the UP.Browser, which supports HDML and WML.

14.2 Sample code for HDML

The sample code for HDML is included in the SG246171.zip as follows:

- This sample UP PvC adapter can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\wap\adapter.pvc\UpPvcAdapter.jar
  <install_path_of_zip>\SG246171\direct\wap\adapter.pvc\UpPvcAdapter.xml
  ```

- The HDML content JSPs can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\hdml\hdml_jsp
  ```

- The content management scripts can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\hdml\scripts\setup_hdml.sql
  ```

14.3 m-commerce direct development for HDML

The development process for m-commerce direct has been defined in 10.2, “m-commerce direct development process” on page 203. The example chapters for creating a PvC adapter, content, etc. used HDML as an example. In this section,we provide the high-level steps required for development.

This chapter makes the assumption that you have met the prerequisites defined in the 10.2, “m-commerce direct development process” on page 203.

Create a PvC adapter for UP (HDML)

Refer to 11.1, “Creating a PvC adapter” on page 206 for more detailed instructions.

Deploy the UP PvC adapter

Refer to 11.2, “Deploying a PvC adapter” on page 218 for details.

Configure the content management

Refer to 12.1, “Content management configuration” on page 226 for details.

Create HDML content JSPs

Refer to 12.2, “Create device-specific content JSPs” on page 232.

Deploy HDML content JSPs

Refer to 12.3, “Deploy content” on page 238.

Test the m-commerce store for HDML

Now that you have deployed the HDML device-specific content JSPs, test the store by entering the URL for your store in the UP.Browser simulator.

15

WAP implementation sample

This chapters provides instructions and sample code for implementing m-commerce for WAP mobile devices using WCS V5.1.

The chapter is organized into the following sections:

- WAP toolkits and test clients
- Sample code for WAP
- m-commerce direct development for WAP

15.1 WAP toolkits and test clients

When developing a mobile application, you will need a combination of toolkits, software simulators and real mobile device hardware for testing. We specifically selected a Nokia and UP-based simulator and real hardware to explore unique issues related to the microbrowsers.

We used the following when developing our WAP sample code for m-commerce:

- Nokia WAP Toolkit V2.1 and simulator
- UP.SDK for WAP
- WAP mobile device hardware

15.1.1 Nokia WAP Toolkit V2.1 and simulator

This section provides information on the Nokia WAP Toolkit V2.1 and the accompanying simulator. One of the many benefits of using the Nokia WAP Toolkit V2.1 for WAP development is that it can be used with a WAP gateway to better simulate testing.

Download

The Nokia WAP Toolkit V2.1 and simulator can be found at:

`http://forum.nokia.com/wapforum/main`

If you are not already registered, you will be required to do so before logging on and proceeding to the download of the toolkit.

We downloaded the following:

- Nokia WAP Toolkit V2.1
- Nokia WAP Toolkit V2.1 - Users Guide
- Nokia WAP Toolkit V2.1 - Developer's Guide
- Nokia 7110 Mobile Handset Simulator January 2000 (optional)

  This download is optional for our example. We used the Blueprint simulator that comes with the toolkit.

Install the toolkit

The Nokia WAP Toolkit V2.1 requires installation of the Java Runtime Environment (JRE) 1.3 (or higher) software. If you do not already have this installed, the toolkit will automatically install it for you.

Refer to the Nokia WAP Toolkit V2.1 - Users Guide for instructions on installing the toolkit and simulators.

Configure the toolkit simulator

Configure the toolkit simulator as follows:

- Configure the Use HTTP tab (since we are working in an emulated environment, we do not need to go through a WAP gateway)
- Use HTTP cookies
- Do not use HTTP authentication
- Do not use Fast Encoding
- Set the suitable proxy for your location (if needed)

15.1.2 UP.SDK for WAP

OpenWave Phone.com, formally Unwired Planet, provides a Software Developers Kit (SDK) that includes a simulator that supports WAP content by its microbrowser, examples, and supporting documentation.

Although Nokia is dominant in Europe, the UP Phone.com browser is used by many phone manufacturers.

There are currently two versions of the UP.SDK that support WAP content.

- UP.SDK V3.2 for WAP
- UP.SDK V4.1 for WAP

Download and install UP.SDK for WAP

You can download the UP.SDK V3.2 for WAP from the following URL:

`http://developer.openwave.com/download/index.html#sdk`

15.1.3 WAP mobile device hardware

In our hardware testing environment, we used two different mobile devices. The selection of the WAP phones (R380 PDA) was done partially based on what was available and supported by the only WAP service provider in the Raleigh area (Cingular). The devices were:

- Ericcson R380

  This mobile phone/PDA hybrid may be one of the best WAP devices available on the market. We used this model in light of its capability (PDA type screen size) and the standard WAP spec browser.

- Motorola V.Series V2282

  This phone allowed us to test with a 4 line of text UP browser.

When using the real WAP mobile phones, we were able to connect to Cingular, dial our Network Access Server (NAS) and configure the phone to use our WAP gateway (IBM Everyplace Wireless Gateway configured for WAP).

In addition to the benefits mentioned, real hardware testing allows you to appreciate usability issues (screen size, formatting, network performance).

15.2 Sample code for WAP

Included in the SG246171.zip file are the following code samples:

- Sample WAP PvC adapter

  This sample WAP PvC adapter has been developed to look for the user-agent WAP, Wap, and Nokia found in the HTTP header. In our example, we used the Nokia WAP Toolkit V2.1 simulator. Additional user agents can be added to fit your needs.

  The source code can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\wap\adapter.pvc\WapPvcAdapter.jar
  <install_path_of_zip>\SG246171\direct\wap\adapter.pvc\WapPvcAdapter.xml
  ```

- Sample WLM content JSPs

  The sample JSPs for product and category display have been tested.

  The WML content JSPs can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\wap\wml_jsp
  ```

  In addition, we have included WLM JSPs converted from the HDML sample for all the JSPs; however, they have not been debugged and tested. They can be found in the following directory with _WML appended to the filename:

  ```
  <install_path_of_zip>\SG246171\direct\wap\wml_jsp\notest
  ```

  They are available without support, as is (as are all the samples).

- Content management scripts can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\wap\scripts\setup_wml.sql
  ```

15.3 m-commerce direct development for WAP

The development process for m-commerce direct has been defined in 10.2, “m-commerce direct development process” on page 203. In this section, we have listed the high level steps and key issues related to creating a PvC adapter for a WAP mobile device.

This chapter assumes that you have met the prerequisites defined in the 10.2, “m-commerce direct development process” on page 203.

Creating a PvC adapter for WAP

Refer to 11.1, “Creating a PvC adapter” on page 206 for detailed instructions.

The sample WAP PvC adapter included in the sample zip file offers some improvements over the UP PvC adapter documented in 11.1, “Creating a PvC adapter” on page 206. By simply changing the User-Agent information for your environment, you can start using WAP by deploying the sample WAP PvC adapter.

The sample PvC adapter for WAP is currently written to check for the following user agent information:

- WAP
- Wap
- Nokia
- UP/4

Example 15-1 displays the sample Java source code for the PvC adapter for WAP mobile devices. This Java code can be imported into VAJ from WapPvcAdapter.jar.

Example 15-1 WapPvcAdapter Java source

```
package com.ibm.commerce.pvcadapter.sample.wap;

public class WapPvcAdapter extends com.ibm.commerce.pvcadapter.PVCAdapterImpl {
/**
 * checkDeviceFormat method comment.
 */
public boolean checkDeviceFormat(javax.servlet.http.HttpServletRequest req,
   com.ibm.commerce.datatype.TypedProperty arg2)
{



String agent = req.getHeader("user-agent");
```

```
   java.util.Enumeration agentList = acceptedUserAgents.elements();
   boolean isAcceptable = false;
   String current = "";

   if (agent != null)
   {
      while (agentList.hasMoreElements() && (!isAcceptable))
      {
         current = (String) agentList.nextElement();

         // the device is accepted only if 'current' is a substring of 'agent'
         isAcceptable = (agent.indexOf(current) > -1 );
      }
      if (isAcceptable) deviceModel = current;
   }
   return isAcceptable;
}
/**
 * getDeviceModel method comment.
 */
public String getDeviceModel() {
   return deviceModel;
}
/**
 * getTerminalId method comment.
 */
public String getTerminalId() {
   return getRequest().getSession().getId();
}
/**
 * Insert the method's description here.
* @return boolean
 * @param sub java.lang.String
 * @param target java.lang.String
 */
private static boolean isSubstring(String sub, String target) {

   if ((sub != null) && (target != null))
   {
      if (target.indexOf(sub) > -1)
         return true;
      else
         return false;
   }
   else return false;
}

   private java.util.Vector acceptedUserAgents = new java.util.Vector();
   private String deviceModel = "";
```

```
public WapPvcAdapter()
{
   super();

// Initialization of the "acceptedUserAgent" vector
   this.acceptedUserAgents.add("WAP");
   this.acceptedUserAgents.add("Wap");
   this.acceptedUserAgents.add("Nokia");
   this.acceptedUserAgents.add("UP/4.");
}

/**
* @return boolean
 */
public boolean httpsRedirection() {
   return false;
}
}
```

Deploy the WAP PvC adapter

Refer to 11.2, “Deploying a PvC adapter” on page 218 for details.

Configure the content management

Refer to 12.1, “Content management configuration” on page 226 for details.

Create WML content JSPs

Refer to 12.2, “Create device-specific content JSPs” on page 232.

Note: The WML content JSPs were created by using the HDML equivalent as a starting point. We used the UP.SDK documentation for converting HDML to WML. All the JSPs have been converted to WML; however, we only debugged and tested the navigation of the JSPs for category and product display.

We included the remaining untested JSPs (notest directory) to save time for anyone who may want to use them as a starting point.

All code is shipped as is and includes no support.

We referenced the following information in our conversion to WML content JSPs:

- Professional WAP, Charles Arehart, et al
- Nokia WAP Toolkit 2.1 Developer's Guide
- UP.SDK 3.2 for WML documentation
- UP.SDK 3.2 for HDML documentation

Deploy content WML content JSPs

Refer to 12.3, "Deploy content" on page 238.

Test m-commerce store for WAP

Now that you have deployed the WAP WML device specific content JSPs, test the store by entering the URL for your store in the WAP simulator.

We tested with the following:

- UP 4.1 Toolkit and simulator
- Nokia WAP Toolkit V2.1, Blueprint simulator
- Ericsson R380 WAP phone/PDA using our EWG WAP Gateway.

16

Palm implementation sample

In this chapter, we provide implementation, development and design guidelines for Palm HTML and Web Clipping-based browsers. In addition, we provide sample code for the Palm HTML browser.

We did not include WAP based browsers because the development of m-commerce for WAP mobile devices is already covered in Chapter 15, “WAP implementation sample” on page 271. When using a WAP browser on the Palm, the Palm takes on the properties of a WAP device for Internet access.

The chapter is organized into the following sections:

- Palm HTML browser implementation
- Palm Web Clipping implementation
- Where to find more information

16.1 Palm HTML browser implementation

This section provides implementation guidelines for designing, developing and testing Palm HTML browser PDAs. The section is organized as follows:

- Design guidelines
- Development guidelines
- Palm development tools
- Sample code

16.1.1 Design guidelines

This section highlights the key issues for designing Palm HTML-based content for m-commerce.

Page structure

The page structure is usually modular and fairly redundant, as seen in Figure 16-1.

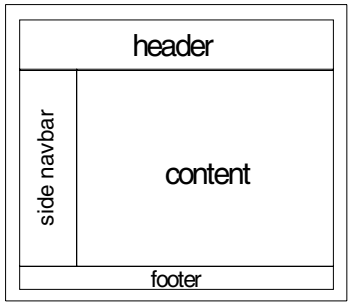


Figure 16-1 Page structure

The above result can be achieved using frames or tables. Frames are usually not supported on PDA browsers, so try to avoid them. First, the structure should be rearranged, considering that the screen is small and narrow. Using sidebars or any horizontal separation is not recommended. Bear in mind that the page content should be linear.

In this case, the side navigation bar, the header, and the footer are playing the same role: site navigation. One approach in modifying the page structure for m-commerce is to simply remove the side navigation bar, modify the header, remove the navigation elements, and simplify the remaining footer.

The main parts (header, content, footer) are separated by horizontal lines using the <hr> tag. By using a linear format, users can read through the pages and navigate.

Tables

Tables are mostly supported on PDA browsers, but nested tables are usually not. Try to avoid using tables. Vertically oriented tables are practical using one or two columns: these are most likely lists. But tables should be avoided for more complex formatting.

In this example, tables are fully removed, the horizontal tables are converted into vertical listings, for example the StoreDisplayPage main category listing.

Vertically oriented tables are simply removed and simply converted to listings, for example the CategoryDisplay page's category listing.

Do not forget to use the
 tag while replacing the cell delimiter, so that the content will be separated from other cells' content.

Images

Images are difficult to handle, because they are not rewriteable. Some text on pages is really images; simply remove them and type in the text. The text can be emphasized using the tags.

There are two ways to reduce image size:

- If size does not matter, the <img src= tag allows you to use width and height attributes to specify the image size (KB). The image can be resized if the browser supports this feature at runtime.
- If size matters, images need to be resized using a tool (static size), and have to be stored in a different directory and served from there.

Forms

Forms are useful to control the interaction on the site and to pass parameters or variables to the server or to the pages. Forms are mostly supported by the PDA browsers.

In this example, image links are used to submit a form, with good results. However, JavaScript is used to launch the submit method of the form. PDA browsers cannot run client-side scripts like JavaScript in case the link does not work, so the form cannot be submitted.

The image links have to be removed, and submit input fields have to be entered into the code, between the <form> and </form> tags, with the appropriate title.

Other design considerations

PDA browsers cannot support Cascading Style Sheets (CSS). Simply remove them along with all the style sheet definitions.

16.1.2 Development guidelines

This section include guidelines for developing m-commerce applications for Palm.

Creating a Palm PvC adapter

To develop a Palm PvC adapter refer to 11.1, “Creating a PvC adapter” on page 206. Once the User-Agent value is identified for the Palm device, the PvC adapter development procedure is the same as documented.

Creating the HTML content JSPs

The working example requires an HTML browser on the device. The example shows how to produce simplified content for these devices. The developer needs to consider the bandwidth, small screen size, and the browser’s capabilities.

Testing

There are several levels of testing environments and types of test clients. In this section, we will briefly describe the issues to be considered in testing with different types of test clients.

- Real hardware

  A real PDA browser is best for testing the site; however, it could be slow and waste battery energy. It is recommended that you test the site with as many PDAs as possible; this is best for the final test.

- PC browser

  During the development process, any ordinary desktop browser could be useful (for example Netscape Navigator or MS Internet Explorer). Switch off features like JavaScript and Java, and for more realistic results, simply resize the browser window so it resembles a PDA screen (effective window size: 250 pixel x 350 pixel). Also, consider the font size.

- Emulators

  There are also emulators for developing and testing a wireless application. Emulators and simulators behave like the original device. Their advantages are faster development, low cost, and easier debugging.

16.1.3 Palm development tools

This section provides a summary of the development tools and documentation available for Palm OS. There are three fundamental development components for developing a Palm application:

- ROM
- Palm OS Emulator
- Palm OS SDK

In our example of developing HTML-based Web applications, we will need to download the PALM emulator, freely available on the Internet, and the ROM image, which is licensed (for development purposes, this is also accessible for free).

Downloads

- The Palm OS Emulator can be downloaded from the following URL:

  `http://www.palmos.com/dev/tech/tools/emulator/`

- The ROM image can be downloaded from the following URL, once you have registered as a member of the Alliance program:

  `http://www.palmos.com/dev/tech/tools/emulator/`

- The developer's documentation can be downloaded from the following URL:

  `http://www.palmos.com/dev/tech/docs/`

Configuring the emulator

After installing the software, access the **Settings->Properties** menu and switch on the **Redirect NetLib calls to host TCP/IP** flag; use the right mouse button on the Palm window.

Use any of the listed browsers under the PALM HTTP browser section or any other browser. There are some typical settings for these browsers:

- Proxy: usually, the browsers require a proxy server with a user name and password. The proxy server performs some content transformation and provides a secure connection.
- Cache: Palm browsers always use cache in order to browse in offline mode after downloading the content.

There are other emulators available on the Internet, as well as special development tools for specific devices available from the manufacturer.

16.1.4 Sample code

This section provides the sample code included in the ITSO SG246171.zip file.

Palm HTML content JSPs

The following Palm HTML content JSPs can be used as a reference to build an m-commerce application. We modified the following JSPs to provide browsing capability for the PVCFashion sample.

The JSPs can be found in the c:\temp\sg246171\palm\html directory:

- header.jsp
- footer.jsp
- StoreCatalogDisplay.jsp
- CategoryDisplay.jsp
- topcategory.jsp
- subcategory.jsp
- ProductDisplay.jsp

16.2 Palm Web Clipping implementation

This section provides design and development guidelines for implementing Palm Web Clipping for m-commerce.

This section is organized as follows:

- Design guidelines
- Development guidelines
- Testing
- Problems
- Example
- Conclusion

16.2.1 Design guidelines

Web Clipping applications consist of two major parts:

- Web Clipping application, which is installed on the device
- Server-side application, that returns results pages to the device

The Web Clipping application has to be built using HTML, and installed at the end user onto the handheld device. A Web Clipping application is like a small Web site stored locally; the starting (or index) page usually provides a form, or list of links, which are the gateways to the live data provided by the server.

The result pages (clippings) are returned by the server side application, as a response to the request from the web clipping application. The result pages are written in HTML.

Web Clipping Proxy Server

A key component of making the system work is the Palm Computing Web Clipping Proxy server found in 3Com Corporation's data center. The Web Clipping Proxy Server is responsible for converting the standard Internet protocols and content from a Web page into a form that is tuned for transmission across a wireless network, and for display on a small device.

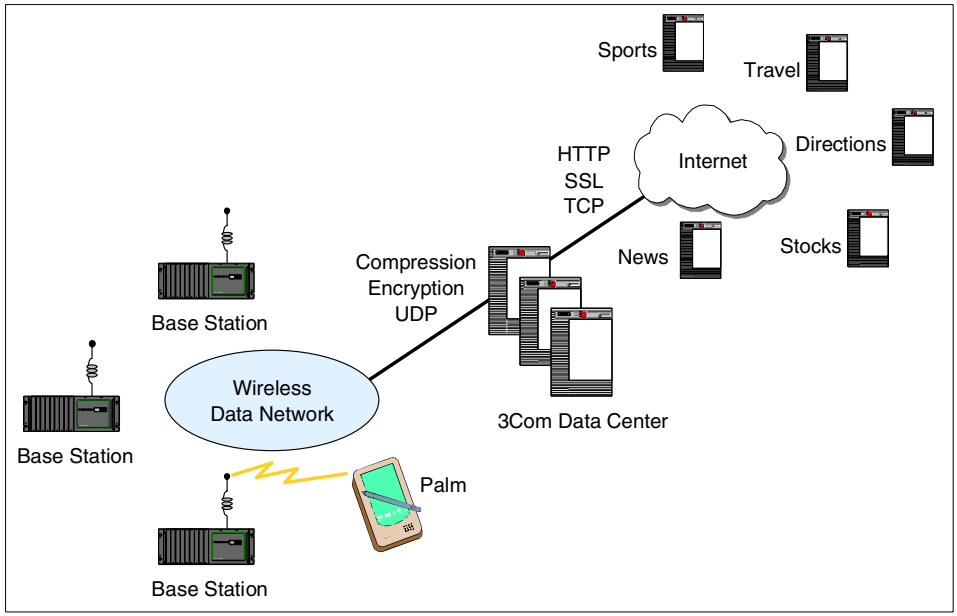


Figure 16-2 Web Clipping Proxy Server

As shown in Figure 16-2, the Web Clipping Proxy uses standard Internet protocols (TCP, HTTP, SSL) to Web servers to ensure compatibility, so the requested pages are HTML pages.

The Web Clipping Proxy server uses a reliable layer over the User Datagram Protocol (UDP) to talk to Palm device; this protocol reduces latency and conserves battery power more so than the Transmission Control Protocol (TCP). Both UDP and TCP are transport protocols within the TCP/IP suite of protocols.

In Web Clipping, encryption and authentication between the handheld device and the Web Clipping Proxy Server is performed by Elliptic Curve Cryptography from Certicom Corporation, which offers extremely high levels of security with small encryption key sizes. On the server side, the high strength SSL is used for encryption and authentication between the Web Clipping Proxy Server and Web servers providing HTML content.

16.2.2 Development guidelines

The process of developing an m-commerce site for Palm using Web Clipping can be divided into the following parts:

- Palm-specific development tools
- Test real hardware on the Internet
- Create and deploy a PvC adapter for Palm devices
- The Web Clipping application, found on the client device.
- The server application, which produces the result pages. Practically, these pages should be JSPs, which are served by the application server.

Additional tools for the development process

The following tools are required for development for the Palm device. They are freely available, and downloadable from `www.palm.com`.

- WCA builder, used to compile the Web Clipping application
- Palm OS Emulator (POSE), used for testing
  - Palm OS ROM file, required to run the emulator

Palm OS viewer

The Palm OS viewer is used by the browser on a Palm for the display of Web pages. There are some limitations to the Palm OS viewer that impact the development process. The viewer supports SSL for secure communication, and forms for Web Clipping queries. The viewer does not support cookies, frames, nested tables, JavaScript, or Java.

Server application

The server application provides the query results for the client. The response is an HTML page, formatted for the Palm device; the restrictions are listed above.

If using WebSphere Commerce Suite V5.1, the response should be developed using JSPs. The JSPs are then compiled into HTML and sent through the Internet, passing through the Web Clipping Proxy Server.

General considerations

Well-designed Web Clipping applications require a balance between the client side application and the server-side application. A normal request sent from the client is a 50-yte data packet, and the response should be 500 bytes. When using images and more content, the size of the data packet may be larger. The ratio of request and response data should remain within the range of 1:10..20.

Note: For more information about Web Clipping application development, refer to: http://www.palm.com, and check Web Clipping Development under Developer and Programming Community.

16.2.3 Testing

There are two approaches to testing the application:

- Using the real hardware device with wireless access: this is an expensive method of testing, and connection time may take long. It is good for beta testing or final testing, but not recommended for the whole development process.
- Using Palm emulator: this is the best method for this job. The emulator behaves just like the original device. There is no charge for the wireless connection, which is faster. There is only one potential problem: if the development machine is sitting behind a firewall, the developer cannot access the Web Clipping Proxy Server. There are ways to solve this problem, for example, using a SOCKS server. Unfortunately, though the request can pass the firewall, the response from the Web Clipping Server cannot come through. The same problem occurs if the application server is sitting behind the firewall. The Web Clipping Proxy has to somehow access the server.

The recommended testing method is to us the emulator; the development environment should be directly connected to the Internet. The final test could be based on real hardware testing.

16.2.4 Problems

Since there is no cookie support in the Palm viewer, URL rewriting is required on the server application.

An SSL connection to the site is made by the Web Clipping Proxy Server. The server only accepts predefined site certificates, which cannot be changed by the end user. During development, the site has to posses a valid site certificate; if it does not, the site will not work. In this case, the e-commerce site is using a secure connection, so the problem cannot be avoided.

16.2.5 Example

Creating a Web Clipping application for the m-commerce site is similar to creating simplified HTML content for a Palm. For details, refer to 16.1, “Palm HTML browser implementation” on page 280. After creating the Web pages for the site, the home page can be used as the base point for Web Clipping.

The high-level steps are as follows:

- Open the first page with the browser (MS Internet Explorer is recommended, because it can save all the content, including images), and save the page.
- Edit the page, make some modifications to fit the Web Clipping application. Resize the pictures, making them smaller.
- Open the WCA builder and the HTML page, then build the Palm Query Application (PQA).
- Install the PQA on the Palm.

This example shows one approach to producing a WCA for Palm. Other m-commerce sites hardcode the whole catalog into one WCA, and the client can browse it locally, with no connection necessary. The challenge is to find the right balance between the client-side and the server-side applications.

16.2.6 Conclusion

Taking every point into consideration, it is very difficult to develop an e-commerce site with Web Clipping. Many serious problems arise when integrating the Web Clipping application with WebSphere Commerce Suite.

Although Web Clipping is now designed to serve in an environment with a low bandwidth, a short battery life, and a small screen, in the near future these limitations will disappear and online browsing will be closer at hand for Palm devices.

16.3 Where to find more information

The best way to find information about wireless devices and applications is to search and browse the Internet. The following Web sites provide some very useful information about Palm OS application development:

- Palm: `http://www.palm.com`
- IBM WorkPad: `http://www.ibm.com/workpad`
- Handspring Visor: `http://www.handspring.com`

17

i-mode implementation guidelines

At the time of writing this redbook, most information regarding i-mode was confidential. A customer must enter into a confidentially agreement with NTT DoCoMo to gain access to the technical information required to develop an i-mode solution. That being said, we want to highlight the significance of i-mode and the efficiency of WCS V5.1 in working with i-mode mobile phones.

WCS V5.1 m-commerce works very well with the i-mode protocol. The m-commerce PvC extensions added to WCS V5.1 were developed and tested specifically for i-mode. In addition, IBM has developed and deployed m-commerce stores in Japan for i-mode mobile phones.

- For information about i-mode, refer to the NTT DoCoMo Web site at:

  `http://www.nttdocomo.com/i/index.html`

- For information about implementing m-commerce solutions using WCS V5.1 for use with i-mode mobile phones, contact IBM Japan at:

  `http://www.ibm.com/jp/contact`

Part 4

m-commerce using WTP implementation

18

m-commerce using WTP implementation and design

In this chapter, we discuss implementation considerations and application design for m-commerce Web site using WebSphere Transcoding Publisher V3.5.

This chapter is organized into the following sections:

- Implementation considerations
- Application design guidelines using WTP

18.1 Implementation considerations

This section provides integration considerations for using WebSphere Transcoding Publisher V3.5 to transcode the content of WCS V5.1.

18.1.1 WTP overview

IBM WebSphere Transcoding Publisher V3.5 has been presented in 5.2, “IBM WebSphere Transcoding Publisher (WTP)” on page 115. This approach to m-commerce offers several benefits for existing Web applications accessible by many types of mobile devices.

Users can access Web services and receive the information in a format that is tailored to mobile devices’ capabilities. Images can be reduced in size or converted into a format that can be displayed on the device. The retrieved content can be converted into a language that the device can handle and display. These types of content adaptations are performed on-the-fly on the basis of profiles that need to be defined only once. IBM WebSphere Transcoding Publisher V3.5 constitutes a single and centralized means of disseminating the same content source to many different devices with various capabilities. Access to the existing content through a new kind of device can be obtained by adding a new profile for the new device. The profile determines how requests for the device can be recognized and how the information retrieved from the Web server should be adapted. The detection of the correct device type is based mainly on the User Agent field of the incoming HTTP request.

The most typical transformations that are performed by IBM WebSphere Transcoding Publisher V3.5 include:

- Converting XML into different markup languages (such as WML, cHTML, HDML, etc.) by applying XSL style sheets.
- Simplifying HTML by various methods, like removing components that are not supported on the target device (for example JavaScript and applets), or converting tables to lists.
- Manipulating images in terms of compression rate, quality, size, etc.

The architecture of WTP also allows you to further customize the content adaptation process by creating and adding new transcoder modules. For detailed information about WTP, refer to the *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965.

WTP 3.5 as a MIME filter

There are two basic ways of deploying IBM WebSphere Transcoding Publisher V3.5:

- Proxy (for example, transparent or reverse proxy with or without external caching)
- MIME filter for the IBM WebSphere Application Server V3.5

The differences between the two deployment options have been described in 5.2, “IBM WebSphere Transcoding Publisher (WTP)” on page 115. We used the MIME filter option (WebSphere Application Server filter) because it is the only approach that allows end-to-end security, which is highly desirable and a requirement of most commerce sites. In an SSL connection, the encryption takes place between the client that has requested the page and the Web server. The MIME filter solution allows the transcoding of the data before it is SSL encrypted by the Web server.

Figure 18-1 shows how the WebSphere Application Server configuration is modified after installing WTP V3.5 as a MIME filter. Three new servlets are added to the Web application that has been selected for transcoding. Among these three servlets, the TranscodingFilter is the actual MIME filter that transcodes the pages produced by the Web application. In the future, we will call this the TranscodingFilter.

.

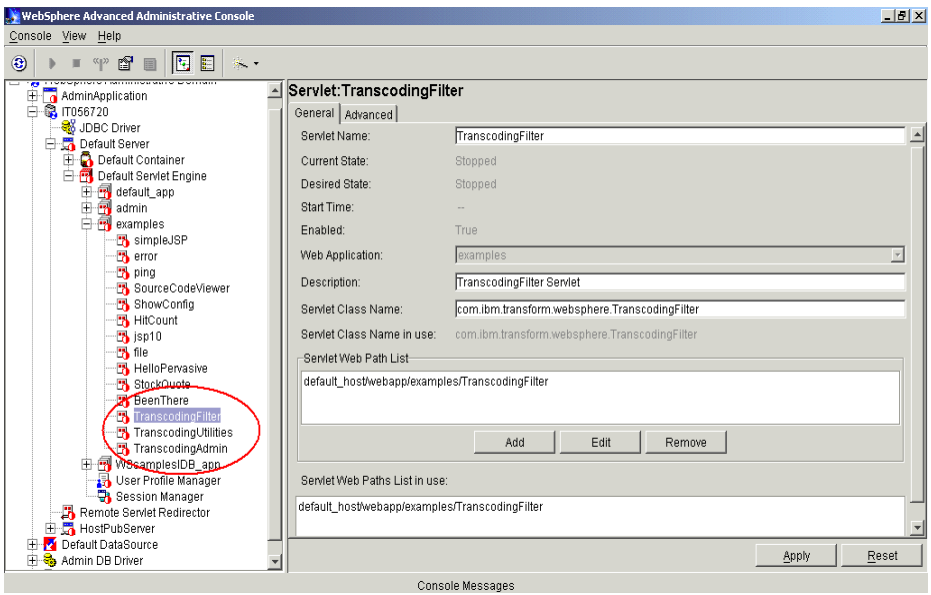


Figure 18-1 Transcoding servlets

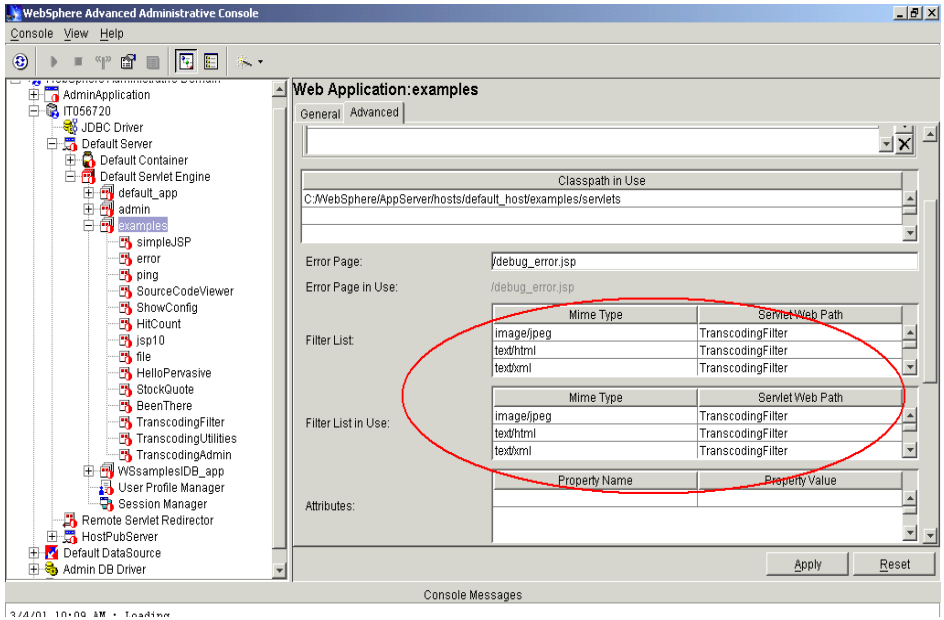


Figure 18-2 Default MIME types

As seen in Figure 18-2, after configuring WTP as a MIME filter the TranscodingFilter servlet is mapped by default to the following MIME types:

- text/html
- text/xml
- image/jpeg
- image/gif

As soon as the Web application generates one of the above content types, the TranscodingFilter is triggered and performs all the content adaptation defined for the client device.

Figure 18-3 shows the complete processing path of an HTTP request directed to a Web application using WTP V3.5 as a MIME filter.

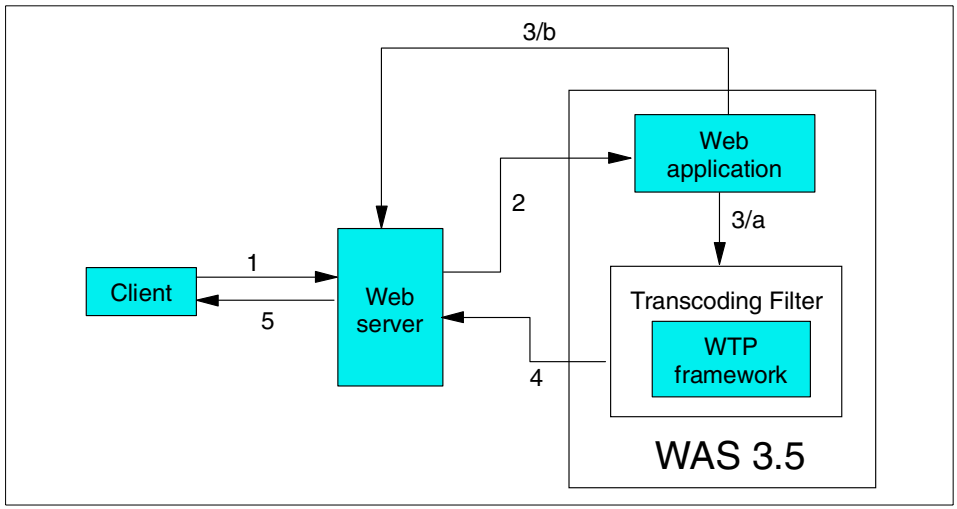


Figure 18-3 WTP 3.5 as a MIME filter

The complete path for any incoming request is the following:

1. The client sends an HTTP(S) request to the Web server.
2. The Web server dispatches the request to the Web application.
3. The Web application generates an output page. Now two options may arise:
   a. The content type of the generated page is associated to the Transcoding Filter servlet as a MIME handler. In this case, the page is passed to the Transcoding servlet.
   b. The content type is not associated to the Transcoding Filter servlet. Then, the page is sent directly to the Web Server.
4. The Transcoding Filter adapts the content on the basis of the preferences for the client device and returns the page to the Web server.
5. The Web server sends the response page to the client.

As seen in Figure 18-3, the Transcoding Filter uses all the services and functionality of the WTP framework, which is the core component of IBM WebSphere Transcoding Publisher V3.5. The WTP framework provides mechanisms to select all the transcoding rules to apply, as well as the execution order.

Note: When WTP is configured as a MIME filter, network profiles are never applied. Only device profiles are applied.

It is also worth noting that only the content that is provided by the application server can be transcoded. If you have static pages that need transcoding, you need to use a folder accessible by the Web application of the WebSphere Application Server (not the Web server document root where static pages are normally stored).

WTP V3.5 includes many transcoders that allow the manipulation and transforming of textual information, such as HTML and XML documents. These transformations include:

1. Simplification of HTML documents.
2. Style sheet (XSL) application to XML documents.
3. Format translation from HTML to:
   a. Wireless Markup Language (WML for WAP mobile devices).
   b. Compact HTML (cHTML for i-mode mobile phones)
   c. Handheld Device Markup Language (HDML mobile devices)

The text transcoder handles the simplification of HTML documents, the XSL application to XML documents, and the translation from HTML to WML. For the other two translations, WTP provides ad-hoc transcoders:

- i-mode transcoder
- HDML transcoder

Note: i-mode and HDML transcoders are not registered by default, but are required to transcode data for i-mode or HDML devices.

If you want to use WTP to translate the pages produced by a Web application into a content format suited to a specific device, two basic options are available as far as the source content format is concerned: HTML or XML. In case the application generates HTML, the Transcoding servlet can be used to convert the page into the proper markup language and eventually to further simplify the content. In case the Web application generates XML, WTP will first apply an XSL style sheet to translate the page into the proper markup language, then will eventually simplify the content.

18.1.2 Architecture for WCS-WTP integration

The overall architecture of a mobile commerce site based on WCS V5.1 and WTP 3.5 is depicted in Figure 18-4. In this case, WCS V5.1 is assumed to generate XML pages.

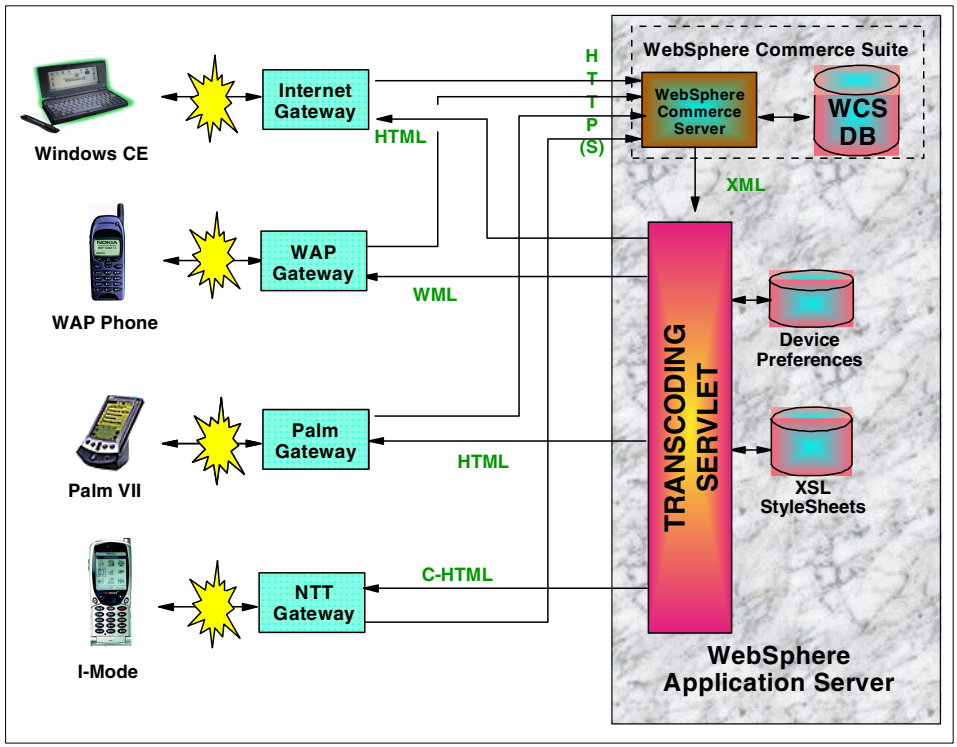


Figure 18-4 Transcoding-based architecture for m-commerce

The existing store infrastructure, based on WCS V5.1, has been extended on the server side by adding the transcoding framework. This is possible because the front ends of both the store and administration tools in WCS are Web applications.

Incoming requests are first processed by the WCS server, which executes the required business logic, then generates an XML output page. The result page is then processed by the Transcoding servlet, which will produce the final output, by first generating the proper markup language and then adapting the content on the basis of the preferences for the client device. The generation of the target markup language is based on the use of proper XSL style sheets.

The style sheet selection is based mainly on the Document Type Definition (DTD) of the retrieved XML page and on the content type supported by the client device. For each kind of device, the transcoding filter knows what style sheet to apply in order to generate the final output.

18.2 Application design guidelines using WTP

When using the WebSphere Transcoding Publisher (WTP) V3.5 approach to mobile commerce, it is important to understand the application design considerations when used with WebSphere Commerce Suite V5.1.

18.2.1 Introduction

The use of WTP to make store pages accessible from mobile devices and, more generally, devices with less capability than common PC browsers is not enough. The content produced by WCS V5.1 is very well suited to PC browsers. It contains JavaScript, applets and other advanced features that are not supported on most mobile devices. Simply transcoding these pages into a different markup language will not do. For example, if the target device does not support JavaScript, WTP will remove this code from the original page. In most cases, WCS uses JavaScript not only for pure layout purposes (for example, to set the focus on a certain component), but also to implement some application logic on the client side (for example, to assign values to fields in a form). If this code is removed, the obtained page will not work properly.

Some further consideration needs to be give to the proper execution of the WCS store. When translating from full HTML, a single page is often split into many contiguous screens (for example, WML decks). The order in which the original information is divided into the different screens is the result of an automatic translation and may not correspond to a logical order. This could have a very big impact on application usability. This aspect could, however, be improved by using Web Clipping.

These considerations show that in order for WTP to work reasonably well, the applications should be carefully designed. There are two major points to keep in mind during the design process:

- Supported features
- Usability

18.2.2 Supported features

HTML can be made efficient on devices with limited capabilities by using WebSphere Transcoding Publisher V3.5. For example, it can properly reduce the number of colors, convert tables into plain text, remove frames, perform image translations or replace images with links, etc. But, as already mentioned, most advanced features can make the translation process very involved. This is the case for image maps, JavaScript, ActiveX objects, Java applets, etc. If the target device does not support this code, these features should not be used.

Most advanced features are not possible to transcode. When there is the need for using such advanced techniques or features that are only supported on some device, you will need to build separate sets of JSPs tailored for different device types.

A compromise can be to build only a few selected pages optimized for specific devices. This could be the home page and pages using incompatible content like Java applets, phone API, Palm software, and so on.

18.2.3 Usability

Designing applications for multiple devices requires considerations about input and output capabilities. Today's common PC Web browsers are very powerful, with a 1024x768 true color screen, a mouse and some 102 keys available. The most minimalistic device used today is the first generation of WAP phones with two to four lines of monochrome text, 10 numeric keys and a few function buttons. In between, you find Web TV, PalmPilot, WorkPad, WinCE, and lots of other devices with various input and output capabilities.

Navigation and application flow clearly need further consideration. On average, 52 links exist on a public Web page. This is not recommended for applications to be used on a phone! An example is a page containing clickable news headlines. The PC Web browser could show 20-30 properly arranged headlines and still be usable. The PalmPilot could probably handle 10 headlines. From a phone, users may prefer only five headlines. Basically, the number of selectable items can be reduced in two ways:

- Categorization

  Introducing one or more category selections in a flow

- Personalization

  Using a filter for passing the subset of relevant items only

The cost and benefit of each approach depend heavily on the application type, content and users. No final answer can be provided. Input forms introduce even more considerations to multi-device design. How many fields per page are acceptable? Does the device support more intelligent forms by allowing client-side scripting? Is text easily entered from the device, or should you rely more on selections? Using the existing user information can be most valuable, for instance by letting the application suggest the customer's known home address when needing a goods delivery address.

18.2.4 Trade-off

You could develop multiple versions of an application, for example, one for large screen users and one for small screen users. Usability can be improved significantly by implementing two different navigation schemes optimized for the two device groups. Device-specific adaptation is performed generically by WTP. This is illustrated in Figure 18-5.

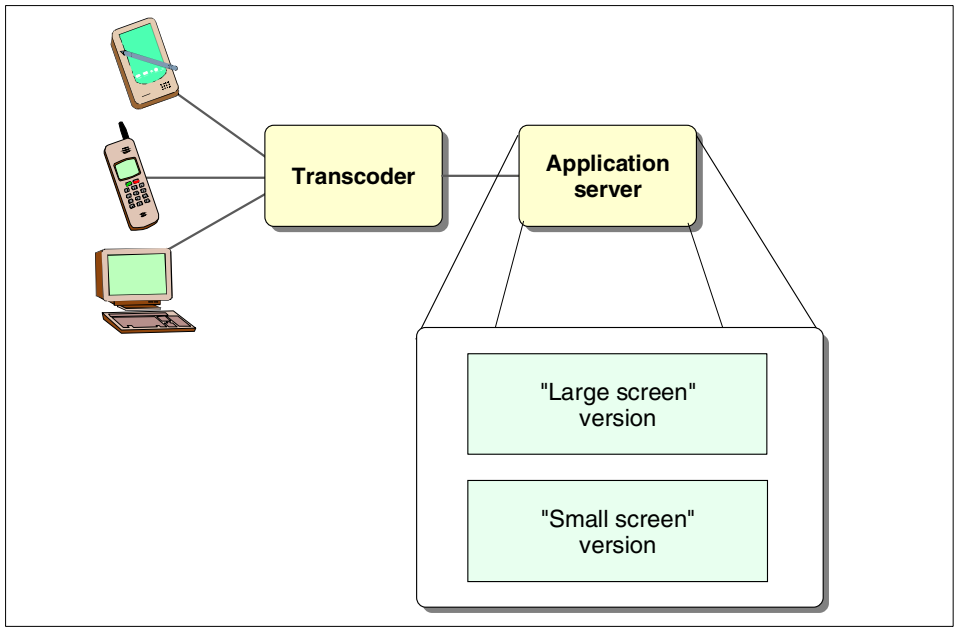


Figure 18-5 Compromise application design

Some devices with a medium-level user interface, like the WorkPad or PalmPilot, could potentially use both versions. The choice can be left up to the user or be determined by the application. As an example of categorization, the application could present a complete menu to a large screen user, while more levels of selection are introduced to the small screen user, allowing each page to be simple. Techniques like personalization could be applied to both versions. The general user interface of both versions should be carefully considered and designed, but by having two specialized versions, less compromises have to be made.

Technically, the application could be implemented as two sets of JSPs: one will generate standard HTML and the other XML or a very simple HTML. These topics are further analyzed in Chapter 19, “m-commerce using WTP application development” on page 305.

19

m-commerce using WTP application development

This chapter provides development guidelines and sample code for developing m-commerce Web sites using WTP. We explore creating HTML and XML content JSPs for WCS, which the WTP server transcodes to the markup language of the target mobile device.

The chapter is organized into the following sections:

- Introduction
- Sample code for WTP
- HTML versus XML
- Selecting the right JSPs
- Creating a generic PvC adapter
- Creating the content JSPs
- WCS caching with WTP

19.1 Introduction

In this chapter, we will assume there already exists a store for PC browsers. We provide a set of guidelines to drive the development flow of a new version suited to mobile devices based on the use of WTP V3.5. The m-commerce direct approach relies on creating a specific mirror of the site for each class of devices. Each mirror produces a content type that is well suited to the requesting device, and the WCS framework relies on proper adapters to uniquely identify the device and select the appropriate content-specific JSPs for that device. This approach is optimal in cases where the set of target devices is limited and well defined. In cases where you want to allow access to as many devices as possible, it may be preferable to have only one reduced version of the store and use WTP to transcode the content on demand for the requesting device.

Before starting the actual development, take note of some important preliminary decisions to guide you in your development:

- Which source markup language to use
- How to map any incoming request to the proper set of JSPs

19.2 Sample code for WTP

Included in the SG246171.zip file are the following code samples:

- Sample Generic PvC adapter

  This sample Generic PvC adapter has been developed to look for the user agent of many types (“WAP”, “Wap”, “Nokia”, “Palmscape”, “Windows CE”, etc.).

  The source code can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\wap\adapter.pvc\GenericPvcAdapter.jar
  <install_path_of_zip>\SG246171\direct\wap\adapter.pvc\GenericPvcAdapter.xml
  ```

- Sample HTML and XML content JSPs for navigation

  The HTML content JSPs can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\wtp\wtp_html
  ```

  The XML content can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\wtp\wtp_xml
  ```

- The content management script can be found at:

  ```
  <install_path_of_zip>\SG246171\direct\wtp\scripts\setup_wtp.sql
  ```

19.3 HTML versus XML

As mentioned in 18.1, “Implementation considerations” on page 296, in order to define a simplified version of an existing store that can be transcoded, both HTML and XML can be used as content types. The following sections provide recommendations that will be helpful, during the development process, in determining which type to use.

19.3.1 HTML content using WTP

The first option is to write a simplified version of the existing store by using an HTML code which is as simple as possible. When writing the JSPs, avoid using elements that may not be supported on the target devices, such as JavaScript, ActiveX, and applets. WTP will remove these advanced components if the target device do not support them. Be aware that the JSPs provided with the InFashion or WebFashion sample stores generate HTML containing JavaScript. This is used not only for layout-specific purposes (for example, to set the focus on one component) but also for application logic (for example, to assign values to fields in a form). If these pieces of code are removed, your store will not work properly.

Some further recommendations follow on the HTML that should be returned by the JSPs:

- Avoid using tables

  Some browsers for wireless devices do not support them and use different data structures. By avoiding tables in the original pages, you will be able to properly arrange the data from the beginning and have a better look and feel of the transcoded content.

- Avoid multiple select fields with the same name

  A problem when writing JSPs producing simplified HTML is related to an implementation technique used within WCS in those pages where many attributes for a certain product are selected and submitted within a form. One typical example is the ProductDisplay.jsp page. Example 19-1 shows simple HTML that uses the same technique.

Example 19-1 Submitting attributes

```
<html>
   <head>
      <title>PvCFashion: Classic pleated dress pant</title>
   </head>
   <body bgcolor="#FFFFFF" text="#000000">
      <form name="OrderItemAddForm" method="get" action="OrderItemAdd">
         <input type="hidden" name="attrName" value="10004">
         <select name="attrValue">
            <option value="Black">Black</option>
```

```
            <option value="Grey">Grey</option>
         </select>
         <input type="hidden" name="attrName" value="10003">
         <select name="attrValue">
            <option value="29W x 28L">29W x 28L</option>
            <option value="30W x 32L">30W x 32L</option>
            <option value="34W x 32L">34W x 32L</option>
         </select>
         <input type="submit" name="AddButton" value="ShoppingCart">
      </form>
   </body>
</html>
```

As you can see, there are two hidden fields with the same name (*attrName*) and two selected fields with the same name (*attrValue*). This is no problem in HTML because all the fields are treated as different entities when the data is submitted. Consider now the page obtained by transcoding from HTML to WML, as seen in Example 19-2.

Example 19-2 Transcoded attributes

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
   <card>
      <p>
         <select name="attrValue" title=" ">
            <option value="Black">Black</option>
            <option value="Grey">Grey</option>
         </select>
         <select name="attrValue" title=" ">
            <option value="29W x 28L">29W x 28L</option>
            <option value="30W x 32L">30W x 32L</option>
            <option value="34W x 32L">34W x 32L</option>
         </select>
         <do name="te2" type="accept" label="ShoppingCart">
            <go href="OrderItemAdd" method="get" accept-charset="ISO-8859-1">
               <postfield name="attrName" value="10004"></postfield>
               <postfield name="attrValue" value="$(attrValue:n)"></postfield>
               <postfield name="attrName" value="10003"></postfield>
               <postfield name="attrValue" value="$(attrValue:n)"></postfield>
               <postfield name="AddButton" value="ShoppingCart"></postfield>
```

```
            </go>
          </do>
          <do name="tel" type="prev" label="Back"><prev/></do>
        </p>
    </card>
</wml>
```

In the obtained WML, the hidden fields have remained distinct, but the two attrValue fields have been mapped to the same value, which is the value chosen in the second select statement.

Note: This particular problem has been reported, but at the time of writing this redbook, a solution was not available.

19.3.2 XML content using WTP

The alternative solution is to write JSPs generating XML. This approach requires more programming than in the previous case, because for each command you will need to:

- Write JSPs generating XML
- Create DTDs for the generated XML
- Create an XSL style sheet for WTP

The DTD is used by the transcoding framework to select which XSL to apply. The XSL is applied to the XML produced by the JSP to generate the proper markup language. This can lead to a very high number of style sheets to create. Suppose your Web site has p JSPs (where p may represent several dozens of JSPs) and you want to allow the access to d different types of devices. In the worst case, you will need d x p style sheets. Sometimes, it is possible to group together similar devices in order to have a same set of XSLs for all of them, and then reduce the number of style sheets to define. However, this requires a great amount of development, not only because of the number of XSLs but also because of the complexity of writing them.

Moreover, WCS already has a clean model-view-controller separation. In this architecture, JSPs are used as the presentation layer. Since XML is not a presentation but a data layer, using JSPs to produce XML is a way of forcing this environment, because we are adding a new presentation layer (the XSLs) on top of the JSPs. The overall framework has been built in such a way that all the presentation-related information is only accessible to the JSPs. This situation can introduce difficulties in writing the JSPs generating XML and the corresponding XSL (for example, JSPs in WCS that extract strings from a

resource bundle for display). There is no means for an XSL to access the same resource bundle to get the text to display and then it is up to the JSP to pass on this information. The XML generated by the JSP will then need to contain not only pure data but also presentation elements.

These difficulties are illustrated in Example 19-3.

Example 19-3 StoreCatalogDisplay: simple XML for the home page

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE StoreCatalogDisplay SYSTEM "StoreCatalogDisplay.dtd">
<StoreCatalogDisplay storeId="10101" langId="-1" catalogId="10001">
   <Menu>
      <Item command="LogonForm" description="Choose country" title="Next">
         <param name="page" value="sidebar"/>
      </Item>
      <Item command="LogonForm" description="Register" title="Next"/>
      <Item command="OrderItemDisplay" description="Shopping cart"
title="Next">
         <param name="orderId" value="."/>
      </Item>
      <Item command="ContactView" description="Contact us" title="Next"/>
      <Item command="HelpView" description="Help" title="Next"/>
      <Item command="PrivacyView" description="Privacy" title="Next"/>
   </Menu>
   <TopCategories>
      <TopCategory id="10001" description="Men's" title="Next"/>
      <TopCategory id="10002" description="Women's" title="Next"/>
      <TopCategory id="10003" description="New Arrivals" title="Next"/>
   </TopCategories>
   <PromoCategory id="24" msg="Check out this month's best-selling products:">
      <PromoProduct id="10032" description="A pair of pants" title="Next"/>
      <PromoProduct id="10022" description="A shirt" title="Next"/>
   </PromoCategory>
</StoreCatalogDisplay>
```

Example 19-3 shows possible XML that could be generated for the store's home page. You will notice that all the strings to use in the final page must be included in this file because only the JSP that generates the page has access to the properties files where the localized information is stored. The XSL can not access the resource bundles to get the same data.

The topmost element in that XML file is called *StoreCatalogDisplay* and has a set of attributes that represents the current context (for example *storeId*, *langId* and *catalogId*).

All the inner elements are grouped into three major categories:

- Menu

  It is the list of functions accessible from the home page. Some of them can have specific parameters, like *page* in the first *Item* element.

- TopCategories

  It is the list of categories that should appear in the home page. Each of them is characterized by an ID and a description.

- PromoCategory

  The PromoCategory represents a current promotion and is identified by a specific ID. It also includes the message used to advertise this promotion on the home page. This category can include many products, each one having an identifier and a description. Example 19-4 shows the DTD for the above XML file.

Example 19-4 StoreCatalogDisplay.dtd

```
   <!xml version="1.0"?>
   <!-- StoreCatalogDisplay.dtd -->
   <!ELEMENT Item (param?)>
   <!ATTLIST Item
   command (ContactView | HelpView | LogonForm | OrderItemDisplay |
PrivacyView) #REQUIRED
      description CDATA #REQUIRED
      title CDATA #REQUIRED
>
   <!ELEMENT Menu (Item+)>
   <!ELEMENT PromoCategory (PromoProduct+)>
   <!ATTLIST PromoCategory
   id CDATA #REQUIRED
      msg CDATA #REQUIRED
>
   <!ELEMENT PromoProduct EMPTY>
   <!ATTLIST PromoProduct
   id CDATA #REQUIRED
      description CDATA #REQUIRED
      title CDATA #REQUIRED
>
   <!ELEMENT StoreCatalogDisplay (Menu, TopCategories, PromoCategory?)>
   <!ATTLIST StoreCatalogDisplay
   storeId CDATA #REQUIRED
      langId CDATA #REQUIRED
      catalogId CDATA #REQUIRED
>
   <!ELEMENT TopCategories (TopCategory+)>
   <!ELEMENT TopCategory EMPTY>
   <!ATTLIST TopCategory
```

```
      id CDATA #REQUIRED
         description CDATA #REQUIRED
         title CDATA #REQUIRED
   >
      <!ELEMENT param EMPTY>
      <!ATTLIST param
      name CDATA #REQUIRED
         value CDATA #REQUIRED
   >
```

One possible style sheet used to generate WML from the above XML file is shown in Example 19-5.

Example 19-5 StoreCatalogDisplay.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">

<xsl:variable name="storeId" select="/StoreCatalogDisplay/@storeId"/>
<xsl:variable name="langId" select="/StoreCatalogDisplay/@langId"/>
<xsl:variable name="catalogId" select="/StoreCatalogDisplay/@catalogId"/>

<xsl:template match="/">
<wml>
   <head>
      <meta http-equiv="Cache-Control" content="max-age=0" forua="true"/>
      <meta name="vnd.up.markable" forua="true" content="true"/>
   </head>
   <xsl:apply-templates/>
</wml>
</xsl:template>

<xsl:template match="StoreCatalogDisplay">
   <xsl:variable name="promoExists" select="count(PromoCategory)"/>
   <card id="DP1">
      <p>
      <img src="/MFashion/pvc/images/logo.wbmp"/>
      <br/>
      MFashion
      <br/>
      <xsl:apply-templates select="Menu"/>
      <xsl:apply-templates select="TopCategories"/>
      <xsl:if test="$promoExists > 0">
            <a href="#DP2"><xsl:value-of select="PromoCategory/@msg"/></a>
      </xsl:if>
      </p>
   </card>
```

```
   <xsl:if test="$promoExists > 0">
      <card id="DP2">
         <p>
            <do type="options" label="Home">
               <go href="#DP1"/>
            </do>
         </p>
         <xsl:apply-templates select="PromoCategory"/>
      </card>
   </xsl:if>
</xsl:template>

<xsl:template match="Menu">
      <xsl:for-each select="Item">
         <xsl:element name="a">
            <xsl:attribute name="href">
               <xsl:value-of select="@command"/>
               <xsl:text>?storeId=</xsl:text>
               <xsl:value-of select="$storeId"/>
               <xsl:text>&amp;langId=</xsl:text>
               <xsl:value-of select="$langId"/>
               <xsl:text>&amp;catalogId=</xsl:text>
               <xsl:value-of select="$catalogId"/>
               <xsl:variable name="paramcount" select="count(param)"/>
               <xsl:for-each select="param">
                  <xsl:text>&amp;</xsl:text>
                  <xsl:value-of select="@name"/>
                  <xsl:text>=</xsl:text>
                  <xsl:value-of select="@value"/>
               </xsl:for-each>
            </xsl:attribute>
            <xsl:attribute name="title">
               Next
            </xsl:attribute>
            <xsl:value-of select="@description"/>
            </xsl:element>
            <br/>
      </xsl:for-each>
</xsl:template>

<xsl:template match="TopCategories">
      <xsl:for-each select="TopCategory">
         <xsl:element name="a">
            <xsl:attribute name="href">
               <xsl:value-of select="CategoryDisplay"/>
               <xsl:text>?storeId=</xsl:text>
               <xsl:value-of select="$storeId"/>
               <xsl:text>&amp;langId=</xsl:text>
               <xsl:value-of select="$langId"/>
```

```
                <xsl:text>&amp;catalogId=</xsl:text>
                <xsl:value-of select="$catalogId"/>
                <xsl:text>&amp;categoryId=</xsl:text>
                <xsl:value-of select="@id"/>
                <xsl:text>&amp;top=Y</xsl:text>
             </xsl:attribute>
             <xsl:attribute name="title">
                Next
             </xsl:attribute>
             <xsl:value-of select="@description"/>
             </xsl:element>
             <br/>
      </xsl:for-each>
</xsl:template>

<xsl:template match="PromoCategory">
   <xsl:variable name="promocatid" select="@id"/>
   <p align="center">
      <xsl:value-of select="@msg"/>
      </p>
      <p>
      <br/>

      <xsl:for-each select="PromoProduct">
         <xsl:element name="a">
            <xsl:attribute name="href">
               <xsl:value-of select="ProductDisplay"/>
               <xsl:text>?storeId=</xsl:text>
               <xsl:value-of select="$storeId"/>
               <xsl:text>&amp;langId=</xsl:text>
               <xsl:value-of select="$langId"/>
               <xsl:text>&amp;catalogId=</xsl:text>
               <xsl:value-of select="$catalogId"/>
               <xsl:text>&amp;productId=</xsl:text>
               <xsl:value-of select="@id"/>
               <xsl:text>&amp;parent_category_rn=</xsl:text>
               <xsl:value-of select="$promocatid"/>
            </xsl:attribute>
            <xsl:attribute name="title">
               Next
            </xsl:attribute>
            <xsl:value-of select="@description"/>
            </xsl:element>
            <br/>
      </xsl:for-each>
      </p>
</xsl:template>
</xsl:stylesheet>
```

The XSL variables initialized at the beginning of the code are used to store the three values that make up the URL of any menu item. This technique avoids having to navigate through the DOM tree any time these three pieces of information are needed.

Note: When writing the JSPs for XML, do not forget to set up the proper content type in the response (for example, text/xml). One possible way is to use the `page` directive, as in the statement:

```
<%@ page contentType="text/xml; charset=iso-8859-1" %>
```

As an alternative you can use the following Java code:

```
<% response.setContentType("text/xml"); %>
```

19.4 Selecting the right JSPs

At the end of the development process, you will have at least two separate sets of JSPs, one producing full-capable HTML for PC browser clients and the other generating simplified HTML or XML for mobile clients. You then need a method to determine which JSP to invoke for any given request.

There are two main options:

Using a PvC adapter

The PvC adapter will allow you to identify the class of devices that should be directed to the simplified pages. You can use a generic adapter to identify all wireless devices that can access your store, without any further distinction. This is needed only to allow WCS to generate different pages for standard PC browsers and wireless devices.

As shown in 12.1, “Content management configuration” on page 226, the PVCDEVSPEC table stores a mapping between the device type detected by the adapter and the document root for that device. As soon as a request coming from a wireless device is detected, the adapter sets the entry point to the store to the directory where all the JSPs (HTML or XML) for that device are stored. The adapter will ignore requests coming from PC browser clients, which will instead refer to the default store location.

The steps needed to deploy a store with mobile support using this approach are:

1. Create a PvC adapter (generic for all wireless devices).
2. Deploy the new PvC adapter.

3. Create the content JSPs for wireless devices in a separate directory from the PC browser client JSPs.
4. Deploy the store assets (possibly by using a SAR).
5. Perform the content management to update the WCS database tables (update the PVCDEVSPE*C* table) by entering the document root for the new set of JSPs.
6. Deploy WTP V3.5 as a MIME filter and configure the transcoders.
7. Restart WebSphere Commerce Server.

Using different stores IDs

If you have two separate stores, they will be accessed through different store IDs and different URLs. This means there is no need for adapters because the distinction between the two stores is based on the different URL and not on the device type. This is the simplest solution to deploy, but it presents additional problems in terms of ease of maintenance and administration. You now have two different stores to manage instead of one. Changing any store settings now requires twice the effort, because any change needs to be applied twice.

To deploy the two stores, you will need to:

1. Create the JSPs for mobile devices.
2. Define two different SARs that only differ in the store file assets. The first contains all JSPs producing full HTML for PC browsers, and the second contains the JSPs generating simple HTML or XML.
3. Publish the two SAR files on the same WCS instance.
4. Deploy WTP V3.5 as a MIME filter and configure the transcoders.
5. Restart WebSphere Commerce Server.

19.5 Creating a generic PvC adapter

This section describes how to develop a generic PvC adapter for mobile devices used with WTP. The technical aspects related to adapters have already been presented in 10.1, “m-commerce direct application design guidelines” on page 200.

19.5.1 PvC adapter overview

The PvC adapters provide the following high-level functionality:

- Detect the device type

- Uniquely identify the device
- Create and maintain the session
- Set up the document root for the targeted device

  The reason why we need an adapter is that we want the store to be able to generate different response pages based on the type of requesting device. More precisely:
- Full HTML for PC browsers
- Simple HTML or XML for small devices

Figure 19-1 depicts how each incoming HTTP request is processed, depending on whether it comes from a pervasive computing mobile device, such as a mobile phone, or from a PC browser.

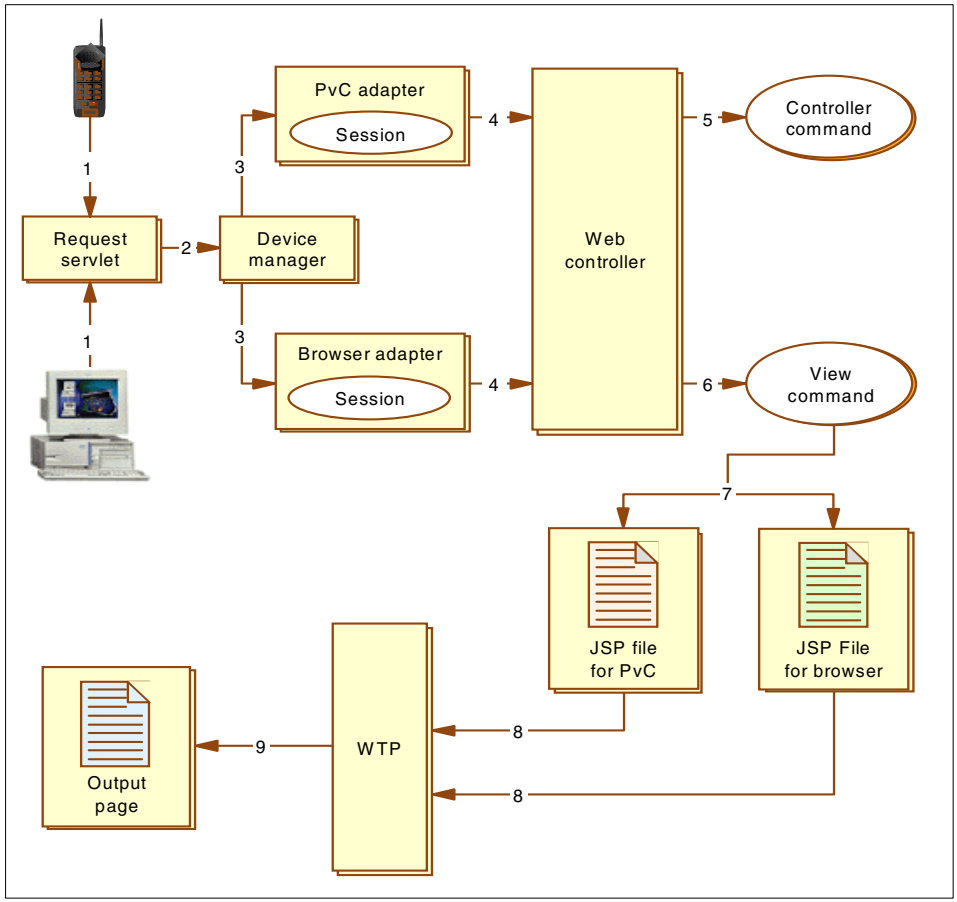


Figure 19-1 Processing flows using WTP V3.5

The processing flow is as follows:

1. A request servlet receives a request from either a mobile device or a PC browser client over the Internet.
2. The request servlet passes the request to a device manager.
3. The device manager determines which adapter would best process the request, and passes the request to the appropriate adapter. For example, if the request is from a mobile phone, the device manager selects the PvC adapter. If the request is from a PC browser, the device manager selects the browser adapter.
4. To prevent applications from having to handle system functions, such as access control and authentication, requests from any device are first processed by the Commerce Suite Web controller. The adapter (either PvC or browser) creates a session context and a controller request object, and passes the controller request object to the Web controller. The controller request object contains a set of properties, formatted by the adapter. It also contains a backward reference to the adapter object and a reference to the session context object created by the adapter.
5. The Web controller executes the request by invoking the corresponding controller command. All business logic is implemented in the controller command.
6. Using the view name returned by the controller command, the Web controller retrieves the appropriate view entry from the VIEWREG table. The WCS framework also constructs the proper JSP file path. If the request comes from a PvC device, it will be obtained by appending the file name in the VIEWREG table to the value of CONTENTDIR in the PVCDEVSPEC table. Otherwise, the file path will be constructed from the default document root.
7. The Web Controller invokes the view command defined in the view entry. Based on the computed path, the view command can fetch the proper JSP from the appropriate folder for the requesting device. If the request comes from a PvC device, the view command may invoke a JSP generating a page written in simple HTML or XML. If the request comes from a PC browser, the view command may invoke a JSP file generating normal HTML (that is, with JavaScript, applets, etc.).
8. The generated page is sent to IBM WebSphere Transcoding Publisher V3.5 in order to further adapt the retrieved content to the capabilities of the target device. In the case of a PC browser, the page may be left unchanged. But if the request comes from a cellular phone, WTP will automatically generate a version of the page whose content format is tailored for that particular device (for example WML, cHTML, etc.).
9. The page generated by IBM WebSphere Transcoding Publisher V3.5 is sent to the requesting device to be displayed.

Note: The document root containing all JSPs for PvC devices is configured in the PVCDEVSPEC table.

19.5.2 Creating the PvC adapter

We will create a new adapter called GenericPVCAdapter. The requesting device can be identified based on the User-Agent field in the HTTP request. The adapter will use the following two private fields:

Table 19-1 Private fields of the GenericPVCAdapter

acceptedUserAgents	It is the set of user agents accepted by the adapter.
deviceModel	It stores the model of the accepted device.

These fields are declared as follows:

```
private java.util.Vector acceptedUserAgents = new java.util.Vector();
private String deviceModel = "";
```

The list of user agents must be further completed with all the accepted values. This can be done in the class constructor.

Example 19-6 GenericPVCAdapter: Class constructor

```
/**
 * GenericPVCAdapter constructor comment.
 */
public GenericPVCAdapter() {
   super();

   // Initialization of the "acceptedUserAgent" vector
   this.acceptedUserAgents.add("Windows CE");
   this.acceptedUserAgents.add("HandHTTP 1.1");
   this.acceptedUserAgents.add("EudoraWeb");
   this.acceptedUserAgents.add("Palmscape");
   this.acceptedUserAgents.add("UP.Browser");
   this.acceptedUserAgents.add(" UP/4.");
   this.acceptedUserAgents.add("WAP");
   this.acceptedUserAgents.add("Wap");
   this.acceptedUserAgents.add("Nokia");
}
```

The code in Example 19-6 detects the following classes of devices:

- Windows CE/Pocket PC
- WML devices

- HDML devices
- IBM WorkPad/Palm

This set can be modified as needed. For example, if you also want to recognize i-mode devices, you can add the following line:

```
this.acceptedUserAgents.add("DoCoMo/1.0/");
```

The main method of the adapter is checkDeviceFormat, as follows:

```
/**
 * checkDeviceFormat method comment.
 */
public boolean checkDeviceFormat(javax.servlet.http.HttpServletRequest arg1,
com.ibm.commerce.datatype.TypedProperty arg2) {

   String agent = req.getHeader("user-agent");
   java.util.Enumeration agentList = acceptedUserAgents.elements();
   boolean isAcceptable = false;
   String current = "";
   if (agent != null)
   {
      while (agentList.hasMoreElements() && (!isAcceptable))
      {
         current = (String) agentList.nextElement();
         // the device is accepted only if 'current' is a substring of 'agent'
         isAcceptable = (agent.indexOf(current) > -1 );
      }
      if (isAcceptable) deviceModel = current;
   }
   return isAcceptable;
}
```

The code first gets the user agent value from the HTTP request and then scans the list of accepted user agents. The device is accepted only if the received user agent contains one of the strings in *acceptedUserAgents*. If the device is accepted, the matching substring is used as a device model to further identify the device in the sequel.

The getDeviceModel method is invoked by the system to get the information about the device model. It can be defined as follows:

```
/**
 * getDeviceModel method comment.
 */
public String getDeviceModel() {
   return deviceModel;
}
```

The next method to define is getTerminalId. Defining such a method for a class of devices with different features and capabilities can be difficult for the following reasons, both related to the current implementation of the PVCAdapterImpl class:

1. The session management is handled by the adapter.
2. For session management, the adapter requires a subscriber unique ID. This unique ID is assigned by the service provider and does not change.

Many devices do not send a unique subscriber ID in the HTTP request; many do not use this concept at all. This unique subscriber ID can be provided by the gateway of the service provider. This is the case for NTT DoCoMo i-mode, and for any provider using UP.Link Server as a WAP gateway. One piece of information that can be used to uniquely identify the subscriber is his/her phone number. This is known by the NAS used to access the network (that is, the node into which the user dials). Some WAP gateways have the capability of interrogating the NAS to get this information. For example, the IBM Everyplace Wireless Gateway V1.1.4 has a built-in component with this capability, called WAP device resolver. EWG 1.1.4 also provides a subscriber ID in the HTTP request. However, this is not a standard feature and we should not assume a subscriber ID always exists.

Accordingly, we will not use the subscriber ID in the GenericPVCAdapter. One possible solution consists in using WAS session IDs as unique identifiers, as follows:

```
/**
 * getTerminalId method comment.
 */
public String getTerminalId() {
   return getRequest().getSession().getId();
}
```

The above code returns the session ID assigned to the current HTTP request by WAS. Session management in WebSphere Application Server *must* then be enabled.

Important: You cannot use WCS session IDs to identify the users. In fact, PvC adapters expect subscriber IDs to be static, at least during the same session. This is the case for WAS session IDs, but WCS session IDs change at every logon.

It is also worth noting that the above code can be used both with cookies and URL rewriting (see 3.1.2, “Session management” on page 72).

With this technique, a new record is inserted in the PVCSESSION table for any new WAS session. WAS sessions start when the user accesses the site and expire when the browser is closed. However, there will be no problem with the large number of entries that will be created.

Since the *PVCSESSION_ID* field uses BIGINT(2^64) columns, you can store up to 0x7FFFFFFFFFFFFFFF records in this table. If there were 10,000,000 sessions / day for the WCS server, you will be able to use it for 2,526,951,242 years: 2^64 records / 365 days / 10,000,000 sessions = 2,526,951,242 years.

However, the KEYS table is not configured correctly, because its current schema does not allow to create PVCSESSION_IDs up to 0x7FFFFFFFFFFFFFFF. In order to support that number of entries, you should use the following SQL statement to set the UPPERBOUND:

```
update keys
   set upperbound = 9223372036854775807
where tablename='pvcsession'
```

19.5.3 Deploying the PvC adapter

Once the adapter has been created, you will need to deploy it in the WCS runtime environment as follows:

1. Select the class within the VAJ Workbench, right-click and choose **Export** from the pop-up menu.
2. In the following window, leave all the default settings, and click **Next**.
3. In the next window, enter the complete path of the lib directory in the WCS installation directory (for example, `<wcs_install_path>\lib`).

   This is the folder where required Java classes are searched for by WCS.
4. Then choose **Finish** to export your code.

Modifying the WCS configuration file

The new adapter needs to be registered in the configuration file of the current WCS instance:

<WCS_INSTALL_DIR>\instances\instance_name\xml\instance_name.xml

The following example shows how the configuration file can be modified. For a detailed explanation of the different parameters please refer to “PvC adapter definition for the WCS instance XML file” on page 330.

```
<HttpAdapters>
   <HttpAdapter name="GenericPVC" deviceFormatId="-2"
deviceFormatType="PVCAdapter" deviceFormatTypeId="-1"
factoryClassname="com.ibm.commerce.sample.adapters.GenericPVCAdapter"
enabled="true">
         <PVCAdapter registrationMode="0" preferredLogonTimeout="20"
bufferTimeout="5"/>
      </HttpAdapter>
</HttpAdapters>
```

Important: Since we are using WAS session IDs to identify the user, 0 is the only available value for the registrationMode attribute in the XML configuration file. This is because WAS session IDs are changed whenever any new session is created. Therefore, it is not possible for WCS to recognize the same user in different sessions. Being capable of recognizing the user over different sessions is fundamental when the registrationMode is set to ‘1’ or ‘2’.

19.5.4 Content management

This section describes the steps necessary for content management configuration.

Modifying the WCS database

The last step in this deployment process is to properly configure the database. Further details on the specific configuration settings can be found in “Content management reference” on page 364. Three tables need to be modified:

- PVCDEVMDL
- PVCDEVSPEC
- PVCMDLSPEC

The following SQL statements can be used to register the adapter information in the database:

```
insert into PVCDEVMDL (MODEL_ID, MODELNAME, SESSIONTYPE )
    values ( -1, '', 'GenericPVC');

insert into PVCDEVSPEC (SPEC_ID, SPECNAME, SESSIONTYPE, CONTENTDIR)
    values (100, 'Generic PVC Default', 'GenericPVC', 'wtp_html' );

insert into PVCMDLSPEC (MDLSPC_ID, STOREENT_ID, MODEL_ID, SPEC_ID)
    values (100, 0, -1, 100);
```

The second statement also sets the document root for all wireless devices to the folder called wtp_html.

Restarting WebSphere Commerce Server

The new settings will be available only after WebSphere Commerce Server is restarted. Then, open the WebSphere Advanced Administration Console, right-click WebSphere Commerce Server and choose **Stop** (see Figure 19-2). After it has stopped, right-click again and choose **Start**.

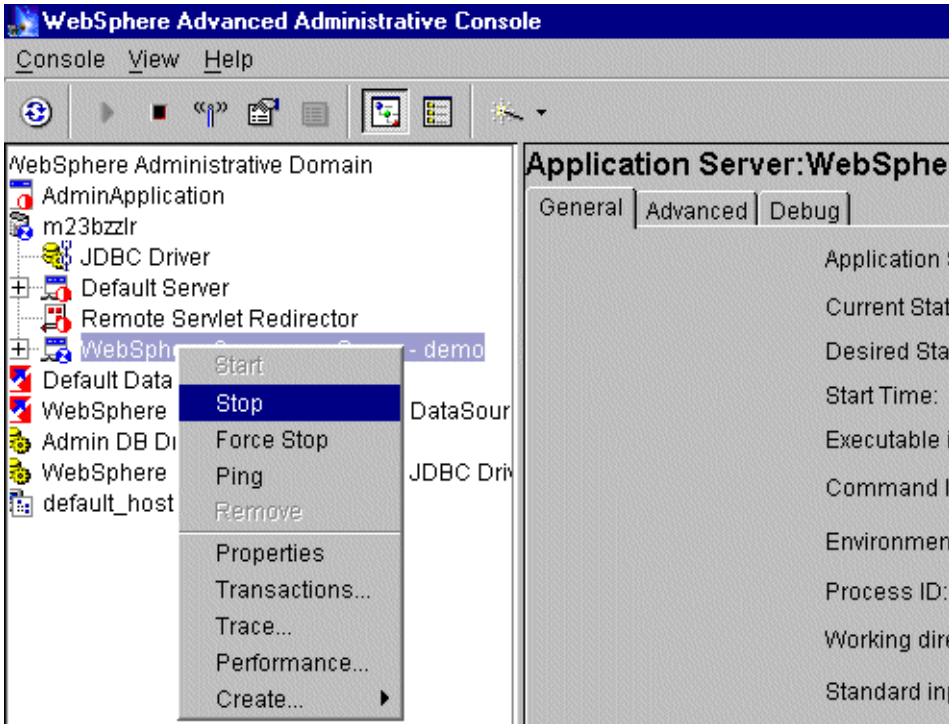


Figure 19-2 Stopping the WebSphere Commerce Server

19.6 Creating the content JSPs

You can create the JSPs for mobile devices either from scratch or simply by modifying the files of your already existing store. You can use WebSphere Commerce Studio for this purpose. In both cases, you will need to create a dedicated folder containing all these new JSPs and all the file assets they need. This new directory must be a subfolder of the store directory. The sample store PvC Fashion will need to be updated with a set of JSPs producing simple HTML

code for wireless devices. Look at the JSPs provided in the additional materials SG246171\wtp\wtp_html directory to understand some of the major guidelines that should frame this development phase. Refer to *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167 for instructions on how to use Studio.

19.7 WCS caching with WTP

If you use WTP V3.5 as a MIME filter for WCS V5.1, it delegates any cache management to WCS. WCS, by default, caches pages on the basis of the URL without any consideration for the device type. This becomes a problem when subsequent different device types access the store and get the transcoded page from the cache.

Consider this situation:

1. You clean up the WCS cache by removing all files in *<wcs_install_path>\instances\<wcs_instance_name>\cache*.
2. You access the store from the WorkPad emulator. You will get simplified HTML tailored for Palm devices.
3. Now, if you try to access the store from a PC browser, even from a different machine, you will get the same simplified version obtained at step 2.

Possible solutions to this problem are:

- Disabling the WCS cache

  This may not be feasible, due to performance, if your Web site has a high volume of clients accessing your site.

- Customizing the cache management in WCS 5.1

  Cache management in WCS V5.1 can be customized. Some caching settings can be customized from the Configuration Manager (as also described in the online help). However, to differentiate the cached pages on the basis of the device type, you need to write your own implementation of the command com.ibm.commerce.cache.commands.CacheCommand. Its getKeyValues method will have to be used to return the values that can uniquely identify the cached pages for the various devices, like the user agent field. However, this has not been tested within our scenarios.

Part 5

Appendixes

A

PvC adapter framework reference

WebSphere Commerce Suite V5.1 provides a PvC adapter framework for providing support of mobile devices in WCS V5.1.

This appendix is organized into the following sections:

- PvCAdapterImpl class and methods
- PvC adapter definition for the WCS instance XML file

PvCAdapterImpl class and methods

The PvCAdpaterImpl class can be found once you have imported the VAJ WCS5101.dat repository included with WebSphere Commerce Suite V5.1 Pro Edition for Windows NT and Windows 2000.

When developing a PvC adapter for your target mobile device type, the following methods must be used:

- checkDeviceFormat

  This method checks to see if the necessary information for session control is contained in the HTTP request from the browser.

- getDeviceModel

  This method is used to get the mobile device model name from the HTTP request.

- getTerminalId

  This method is used to get the unique ID from the mobile device HTTP request.

Note: In the event that your mobile client type or wireless service provider does not supply a unique ID, use the WebSphere Application Server (WAS) session information. For example, we used the WAS cookie as the unique identifier.

PvC adapter definition for the WCS instance XML file

The WCS instance XML configuration file can be found in the following directory of your WCS V5.1 runtime environment:

```
\<wcs_install_path>\instances\<wcs_instance>\xml\<wcs_instance>.xml
```

For example, on a Windows NT WCS V5.1 platform, the file would be:

```
c:\ibm\wcs\instances\wcs\xml\wcs.xml
```

Where `wcs` is the name of the WCS instance.

Example 19-7 provides the entries required for the definition of a PvC adapter. Notice we open with <HttpAdapters> and close it with </HttpAdapters> in the XML configuration file. You must enter this manually since it is not contained in the XML file created by the WCS V5.1 Configuration Manager when creating an instance.

Arrange the tags of HttpAdapters so that it will be an element of the <Config> tag in the XML file. The adapter definition is added within the HttpAdapters tags and uses the syntax as seen in Example 19-7.

Example 19-7 PvC adapter definition syntax for WCS instance XML configuration file

```
<HttpAdapter [HTTP Adapter Attribute] >
         <PVCAdapter [PVC Adapter Attribute]>
       <IPCheck>
          <IP [IP Attribute 1]>
          ...
          <IP [IP Attribute n]>
       </IPCheck>
       <ExcludeCommands>
          <Command name=[[Excluded Command Name 1]/>
          ...
          <Command name=[Excluded Command Name n]/>
       </ExcludeCommands>
       <RelogonCommands>
          <Command name=[Protected Command Name 1]/>
          ...
          <Command name=[Protected Command Name n]/>
       </RelogonCommands>
    </PVCAdapter>
</HttpAdapter>
```

The following tags in brackets provide information about the PvC adapter to be deployed. Each attribute has a table with the possible values.

- [HTTP Adapter Attribute]
- [PVC Adapter Attribute]
- [IP Adapter Attribute 1]
- [IP Attribute n]
- [Excluded Command name 1]
- [Excluded Command Name n]
- [Protected Command Name 1]
- [Protected Command Name n]

HTTP adapter attribute

Table A-1 provides a listing of the HTTP adapter attributes.

Table A-1 HTTP adapter attributes

name	Specify a name for the adapter. It is necessary that this value be unique among all the registered adapters. By all means prevent changing names that have already been registered.
deviceFormatId	Specify a positive integer value that does not overlap other adapters'.
deviceFormatType	Specify 'PVCAdapters' for adapters made after PVCAdapterImpl.
deviceFormatTypeId	Specify -1 for all the adapters made after PVCAdapterImpl.
factoryClassname	Specify the class name of an adapter to be registered.
enabled	It is specified whether this adapter will be used for session control or not. When using, 'true' is specified and when not using, 'false' is specified.

PvC adapter attribute

Table A-2 provides a listing of the PvC adapter attributes.

Table A-2 PVC adapter attributes

registrationMode	0: The connection which is processed by this adapter will be dealt with in the same way as the browser working on the PC. When an ID has already been registered, it can be used from devices controlled by the adapter without any further registration. 1: Registration becomes necessary at logon for devices being processed by this adapter. 2: The device that one user can use is restricted to one unit. When this value is specified, the value that other adapters can specify will be restricted to 0 or 2.
preferredLogonTimeout	The default logon timeout time of this adapter can be set as a countermeasure against loss and theft.
bufferTimeout	The time limit of the effective parameter that has been buffered by the PVCBufferUrl command is specified. Access to the parameter that has been buffered is not permitted after a certain time lapse.

IP adapter attribute

Table A-3 provides a listing of the IP adapter attributes.

Table A-3 IP adapter attributes

type	The address type that is permitted for connection (’host' or 'net') is specified. When 'host' is specified, it means that the configuration permits a connection from an IP address which has the same value. When 'net' is specified, the number of bits set for the value of mask is considered a sub-net mask, and it allows a connection from the designated network address in ‘value'.
value	The IP address of the sender permitting a 'value' connection or the network address is written. It varies in the value specified in 'type' for which meaning it is configured.
mask	Mask is a necessary attribute when specifying 'type= "net"'. Specify the number of bits of the sub-net mask of the network address specified for 'value'. For example, when a sub-net mask is 255.255.255.0, we specify 'mask= "24"'.

Excluded command

The Excluded command is used for a command name that you do not want to execute. This command is used with clients connected with the server through the registration adapter.

Protected command

The Protected command is used to protect the execution by password reentry for the name of the command specified. When a logged-on user tries to execute a specified command without a password, a form is displayed prompting the user to re-enter the password.

Now, let us do these configurations for the UPLinkGateway adapter. The sample configuration settings are listed in Table A-4 through Table A-7.

Sample usage for PvC adapter definition for WCS instance XML file

1. Sample: HTTP adapter attribute

   Information about the adapter to be registered is specified in the HTTP adapter attribute. Specify the factoryClassname of the adapter for the PvC adapter name used when creating the adapter in VAJ (for example, factoryClassname="com.ibm.commerce.sample.adapters.UPLinkGatewayAdapter"). A positive integer that does not overlap with other values is specified in deviceFormatId. Here we use 1, supposing that there is no other adapter registered. As for PVCAdapterdeviceFormatType, deviceFormatTypeId,and PVCAdapter,- 2 is specified. When an adapter is used for session control, we enter “true” as the value of “enabled”. Values for the UPLinkGateway adapter are provided in Table A-4.

Table A-4 Sample HTTP adapter attributes

Item	Configuration value	Note
name	UPG	This is the name of the adapter
deviceFormatId	1	All adapters have unique positive integer value.
deviceFormatType	PVCAdapter	Any adapter which extends PVCAdapterImpl should have this value.
deviceFormatTypeId	-2	Any adapter which extends PVCAdapterImpl should have this value.

Item	Configuration value	Note
enabled	true	To activate the adapter for session control, specify ‘true’.

2. Sample: PVC adapter attributes

   The registration limitation, logon effective time, and time limitation of the buffered parameters can be decided in the PvC adapter attribute. For the meaning of this value, refer to the user registration mode. Before going on to the detailed explanation, we will configure the following.

Table A-5 Sample - PvC adapter attributes

Item	Configuration value	Note
registrationMode	1	Users are required to input some information about their PVC device before logging on to the WCS server.
preferredLogonTimeout	60	The logon is timed out when there is no operation for 60 minutes.
bufferTimeout	10	Users will be prohibited from using the mobile device when the buffered parameter is not updated or changed after 10 minutes.

3. Sample: IP attributes

   The address of the client who permits a connection is specified in the IP attribute. The gateway address of the carrier is entered to prevent forbidden access or forging of the contents of the request. When the address of the client who permits a connection is listed as an IP tag, the address where the list of the IP tags was used will be checked. If an address check functions, it is possible to make session control not function using the adapter when the client address does not correspond to the listed address, even if the request

from the browser was sent in the form that the adapter can handle. An IPCheck tag can be omitted when the WCS server and carrier are connected with leased lines, because it is assured that the request is transmitted by the gateway of the carrier.

Table A-6 Sample IP attributes

	type	value	mask	Note
IP Attribute 1	net	192.168.0.0	24	
IP Attribute 2	host	9.24.105.178		

4. Sample: Excluded command

   We list commands that need to be made unexecutable in the Excluded command. We will set AddressAdd, an AddressDelete command, so that it will not be used from the UPLinkGatewayadapter.

Table A-7 Sample: Excluded command

Excluded command name	Value
Excluded Command 1	AddressAdd
Excluded Command 2	AddressDelete

5. Sample: Protected command

   The OrderProcess command needs to be protected by password reentry when executing using the Protected command.

A summary of all the settings for a sample PvC adapter definition is shown in Example A-1. Insert the definition into the WCS instance XML configuration file within the HttpAdapter tag. When WCS server and the carrier's gateway are connected with leased lines, the IPCheck tag used for gateway address checking can be omitted, since all the requests are transmitted via the regular gateway. For the configuration to take effect, restart the WCS server.

Example: A-1 Sample PvC adapter definition

```
<HttpAdapters>
<HttpAdapter
   name = "UPG"
   deviceFormatId = "-2"
   deviceFormatType = "PVCDevice"
   deviceFormatTypeId = "1"
   factoryClassname="com.ibm.commerce.sample.adapters.UPLinkGatewayAdapter"
   enabled="true" >
```

```
    <PVCAdapter
    registrationMode="1"
    preferredLogonTimeout="60"
    bufferTimeout="10" >
       <IPCheck>
          <IP type="net" value="192.168.0.0" mask="24" />
          <IP type="host" value="9.168.91.113"/>
       </IPCheck >
       <ExcludeCommands>
          <Command name="AddressAdd"/>
          <Command name="AddressDelete"/>
       </ExcludeCommands>
       <RelogonCommands>
          <Command name="OrderProcess"/>
       </RelogonCommands>
    </PVCAdapter>
 </HttpAdapter>
 </HttpAdapters>
```

B

PvC command reference

This appendix is provided as a reference for syntax and usage information for the following PvC commands included in WebSphere Commerce Suite V5.1:

- PVCRegistration
- PVCRegistrationDevice
- PVCChangeDevice
- PVCBufferUrl
- ReEnterPassword

PVCRegistration

This is a command that enables the registration and renewal of the PvC device information of users; registration records of users and PvC device information records can thereby be made and updated. This command is used together with Secure Socket Layer (SSL) to encode the user's logon ID, password and individual information.

This command can be used together with SSL as follows:

- Enter this command by using the HTTPS secure protocol.
- Assign this command to the SSL protocol by using Commerce Suite administrator and the PVCRegistration command (this provides the user's registration information first).

Behavior

1. Checks parameters.
2. Executes the UserRegistration command that performs the user registration and update.
3. Executes the PVCRegisterDevice command that performs the PVC device information registration and update.
4. Displays the specified redirect view.

Commands executed internally

- UserRegistrationAdd
- UserRegistrationUpdate
- PVCRegisterDevice

Tasks command executed internally

- VerifyCredentials

Syntax

The syntax for the PVCRegistration command is shown in Figure B-1.

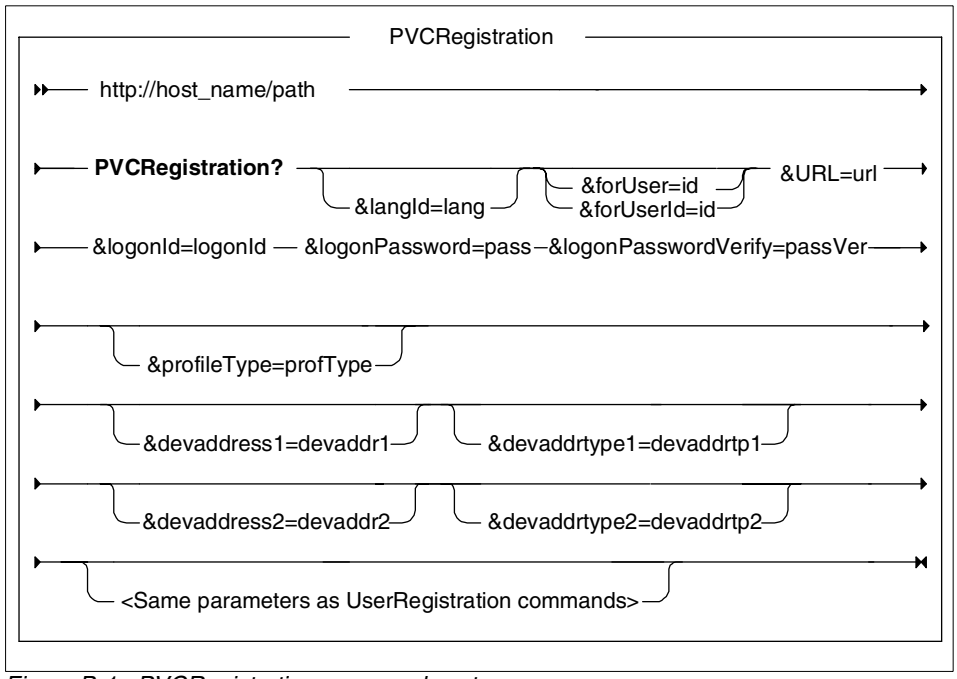


Figure B-1 PVCRegistration command syntax

Arguments

The URL, logonId, logonPassword, and logonPasswordVerify parameters are required. Add the other parameters as needed. When using the Lightweight Directory Access Protocol (LDAP) for authentication mode, the lastName parameter is indispensable.

The arguments and a brief description of each follow:

- http://<host_name>/<path>

  The fully qualified host name of the Commerce Suite server and the configuration path.

- langId

  The language used in this session. Refer to the STORELANG table for the language supported.

- forUser

  The logon ID of the user who executes this command.

- forUserid

  This is the same as the forUser parameter.

- URL

  The URL called when a command finishes without error.

- logonId

  The logon ID used to register and update.

- logonPassword

  The password used to register and update. Passwords are encoded before stored in the database.

- logonPassworVerify

  The password used for confirmation.

- profileType

  The profile type of registration.

  ```
  None: No profile data.
     C: B2C profile type.
     B: B2B profile type.
  ```

- devaddress1

  The PvC information 1 of the user.

- devaddrtype1

  The PvC information type 1 for the user.

- devaddress2

  The PvC information 2 for the user.

- devaddrtype2

  The PvC information type 2 for the user.

Example

The following URL is a sample used to register a new user using the PVCRegistration command”

```
http://myhost/webapp/commerce/store/servlet/PVCRegistration?URL=MallFrontVi
ew&registertype=G&storeId=0&profileType=customer&logonId=user1&logonPasswor
d=password&logonPasswordVerify=password&devaddress1=guest@sample.ibm.com&de
vaddrtype1=email
```

The new user ID and PvC information for logon ID 'user1' is created and calls on the MallFrontView view command.

Error views

The following error views are called when an error occurs.

- UserRegistrationErrorView

  This page is called when an error related to user registration occurs. Parameters related to user registration are not described here. For details, please refer to the WebSphere Commerce Suite V5.1 online help.

- PVCRegistrationErrorView

  This page is called when an error related to client device registration occurs.

Error codes

- ECSecurityConstants.ERR_MISSING_LOGONID

  Logonid is not specified.

- ECSecurityConstants.ERR_MISSING_PASSWORD

  logonPassword is not specified.

- ECConstants.EC_PVC_ADDRESS1

  devaddrtype1 is specified, but devaddress1 is not specified.

- ECConstants.EC_PVC_ADDRTYPE1

  devaddress1 is specified, but devaddrtype1 is not specified.

- ECConstants.EC_PVC_ADDRESS2

  devaddrtype2 is specified, but devaddress2 is not specified.

- ECConstants.EC_PVC_ADDRTYPE2

  devaddress2 is specified, but devaddress2 is not specified.

- ECConstants.EC_PVC_ALREADY_REGISTERED

  This device has already been registered.

- ECSecurityConstants.ERR_INVALID_PASSWORD

  logonPassword for this logonId is wrong.

- ECConstants.EC_PVC_USER_ALREADY_REGISTERED

  The user information of the specified user has already been registered.

- Same error codes as UserRegistrationAdd or UserRegistrationUpdate

  Refer to the WebSphere Commerce SuiteV5.1 online help for error codes returned by UserRegistrationAdd and UserRegistrationUpdate.

Note: The Java classes for ECConstants and ECSecurityConstants are as follows:

```
com.ibm.commerce.server.ECConstants
com.ibm.commerce.security.ECSecurityConstants
```

PVCRegistrationDevice

This command enables the registration and update of PvC information for users that have already registered (for example, from a PC browser client). This command is used together with Secure Socket Layer (SSL) to encode a user's logon ID, password and individual information.

To use this command together with SSL, do the following:

- Enter this command using HTTPS secure protocol.
- Use Commerce Suite Administrator and PVCRegistration command (this displays user registration information first) to assign the command to the SSL protocol.

Behavior

1. When registrationMode equals 0 in the configuration file, it displays the specified redirect view and returns.
2. Checks parameters.
3. Determines the status of the PVCSESSION table, registers when the status is logout, and then updates when the status is logon.
4. For registering, the following is done:
   a. When registrationMode of the configuration file equals 2, it checks whether or not the device which executed the command has already been registered. If so, it displays an error view.
   b. Executes the VerifyCredentials task command to check logon ID and password. In case of an error, it displays an error view.
   c. Retrieves the user ID from the logon ID.
   d. Checks whether the specified logon ID is already registered. If that is the case, it displays an error view.
   e. Creates new PVCBINDING and USERPVCDEV table entries.
   f. Updates the user ID of USERREG and the command context.
5. Does the following when updating:
   a. Updates the USERPVCDEV table of the device that executed the command. It will not update if the parameter itself is not specified.
   b. When the command execution finishes without error, it displays the specified redirection view.

Commands executed internally

- None

Tasks executed internally

- VerifyCredentials

Syntax

The syntax for the PVCRegistrationDevice command is shown in Figure B-2.

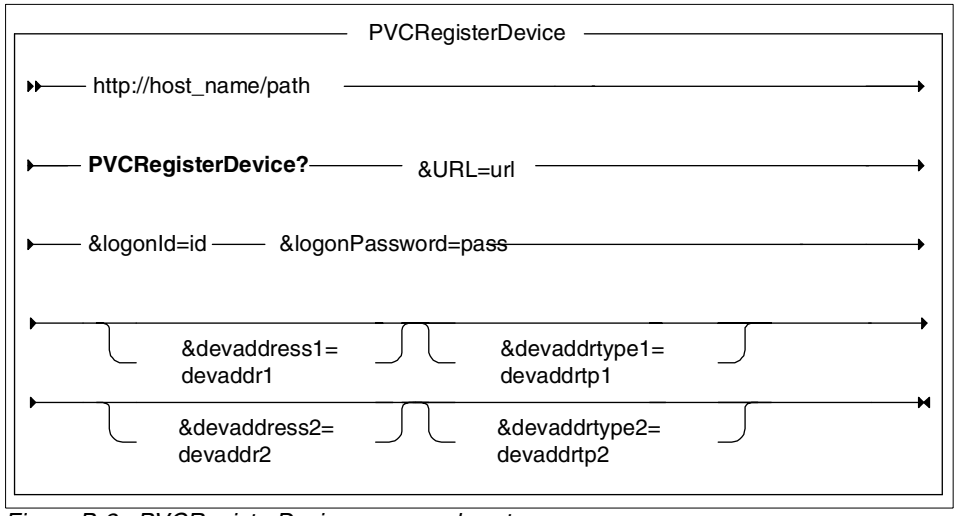


Figure B-2 PVCRegisterDevice command syntax

Arguments

For a new registration (access by a guest user), the logonId and logonPassword parameters are required. For an update (access by a registered user), these parameters are not required.

- http://host_name/path:

  The fully qualified host name of the Commerce Suite server and the configuration path.

- URL

  The URL called when a command is normally terminated.

- logonID

  The logon ID for registration and update.

- logonPassword

  The password used for registration and update.

- devaddress1

  The user's PvC information 1

- devaddrtype1

  The user's PvC information type 1
- devaddress2

  The user's PvC information 2
- devaddrtype2

  The user's PvC information type 2

Example

The following example creates PvC information using the PVCRegistration command for logon ID “user1”, and then calls the MallFrontView view command.

```
http://myhost/webapp/commerce/store/servlet/PVCRegistration?URL=MallFrontVi
ew&logonId=user1&logonPassword=password&logonPasswordVerify=password&devadd
ress1=user1@mobile.ibm.com&devaddrtype1=email
```

the following example updates pVC information using the PVCRegistration command for a user who has already logged on, and then calls the MallFrontView view command.

```
http://myhost/webapp/commerce/store/servlet/PVCRegistration?URL=MallFrontVi
ew&devaddress1=guest@sample.ibm.com&devaddrtype1=email&devaddress2=090-1234
-5678&devaddrtype2=imode
```

Error views

- PVCRegisterDeviceErrorView

Error Codes

- ECSecurityConstants.ERR_MISSING_LOGONID

  logonId is not specified.
- ECSecurityConstants.ERR_MISSING_PASSWORD

  logonPassword is not specified.
- ECConstants.EC_PVC_ADDRESS1

  devaddrtype1 is specified, but devaddress1 is not specified.
- ECConstants.EC_PVC_ADDRTYPE1

  devaddress1 is specified, but devaddrtype1 is not specified.
- ECConstants.EC_PVC_ADDRESS2

  devaddrtype2 is specified, but devaddress2 is not specified.
- ECConstants.EC_PVC_ADDRTYPE2

  devaddress2 is specified, but devaddress2 is not specified.

- ECConstants.EC_PVC_ALREADY_REGISTERED

  This device has already been registered.
- ECSecurityConstants.ERR_INVALID_PASSWORD

  logonPassword for this logonId is wrong.
- ECConstants.EC_PVC_USER_ALREADY_REGISTERED

  The user information of the specified user has already been registered.

PVCChangeDevice

If your site is configured to manage PvC devices as one device per user by setting the session registrationMode to 2 in the XML configuration file, you need to manage canceled or changed devices. This is necessary, because once a user has registered his/her device, no further registration is allowed using another device. The account in the WCS database is locked until the old user-device relationship is canceled.

Some service providers keep a list of revoked subscriber IDs by contract. In this case, you do not need to ask your customer to cancel the old user-device relationship. You can delete the revoked user-device relationship by deleting related records in the PVCSESSION table.

For example:

```
delete from PVCSESSION where
   TERMINAL=REVOKED_SUBSCRIBER_ID
   SESSIONTYPE=NAME_OF_ADAPTER
```

Note: The following values can be used to delete terminals and sessiontype from the PVCSESSION table:

- REVOKED_SUBSCRIBER_ID

  Specifies the revoked subscriber ID which is provided by the carrier.
- NAME_OF_ADAPTER

  Specifies the name of the adapter used for access for the service provided by the carrier.

By deleting a record in the PVCSESSION table, the records in the PVCBINDING table, which stores possible user-device relationship, are deleted. By deleting a record in the PVCBINDING table, the user-device relationship is canceled.

When you are not able to get a list of revoked subscriber IDs and want to use registrationMode 2, you need to ask your customer to cancel the old user-device relationship by typing in the user ID and password using this command. This command is used together with Secure Socket Layer (SSL) to encode the user's logon ID, password and individual information.

To use this command together with SSL, do the following;

- Enter this command using the HTTPS secure protocol.
- Assign this command to the SSL protocol using Commerce Suite Administrator and the PVCRegistration command. (this displays the user's registration information first.)

Behavior

1. When registrationMode is set to other than 2, it indicates a redirect view and returns.
2. Checks that the status of the device that executed the command is logon.
3. Displays an error view when the status is logon.
4. Checks the parameters.
5. Executes the VerifyCredentials task command to check the logon ID and password. In case of an error, it displays an error view.
6. Acquires the user ID from the logon ID.
7. Checks whether or not the device that executed the command has already been registered. If it has, it displays an error view.
8. Checks whether or not the logon ID that is specified has already been registered. If it has, it displays an error view.
9. Changes the status of PVCBINDING of the device that will be the original to S.
10. Creates a new entry in the PVCBINDING table with the status for the device, which is changed to 'A'.
11. Creates new USERPVCDEV of the device that will be changed.
12. When the redirect view is normally finished, displays the specified redirect view.

Commands executed internally

- None

Tasks command executed internally

- VerifyCredentials

Syntax

The syntax for the PVCChangeDevice command is shown in Figure B-3.

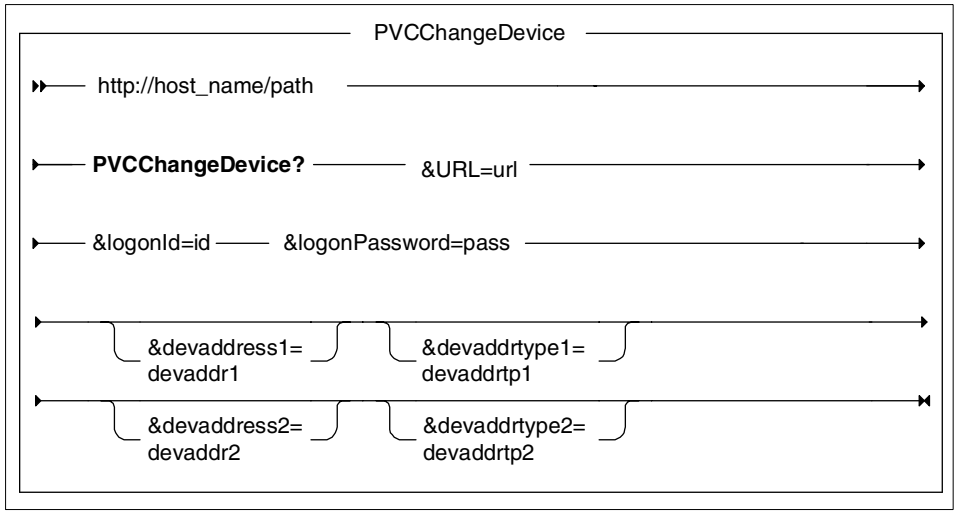


Figure B-3 PVCChangeDevice command syntax

Arguments

- http://host_name/path

  The fully qualified host name of Commerce Suite server and configuration path.

- URL

  The URL called when a command is finished normally.

- logonId

  The logon ID used to register and update.

- logonPassword

  The password used to register and update.

- devaddress1

  The user's PvC information 1

- devaddrtype1

  The user's PvC information type 1

- devaddress2

  The user's pVC information 2

- devaddrtype2

The user's PvC information type 2

Example

The command changes the PvC information of the logon ID "userid", and calls the MallFrontView view command.

```
http://myhost/webapp/commerce/store/servlet/PVCChangeDevice?URL=MallFrontVi
ew&logonId=userid&logonPassword=password&devaddress1=userid@sample.ibm.com&
devaddrtype1=email
```

Error Views

- PVCChangeDeviceErrorView

Error Codes

- ECSecurityConstants.ERR_MISSING_LOGONID

  logonId is not specified.

- ECSecurityConstants.ERR_MISSING_PASSWORD

  logonPassword is not specified.

- ECConstants.EC_PVC_ADDRESS1

  devaddrtype1 is specified, but devaddress1 is not specified.

- ECConstants.EC_PVC_ADDRTYPE1

  devaddress1 is specified, but devaddrtype1 is not specified.

- ECConstants.EC_PVC_ADDRESS2

  devaddrtype2 is specified, but devaddress2 is not specified.

- ECConstants.EC_PVC_ADDRTYPE2

  devaddress2 is specified, but devaddress2 is not specified.

- ECConstants.EC_PVC_ALREADY_REGISTERED

  When executed with registrationMode=2 in the configuration file, this device has already been registered.

- ECSecurityConstants.ERR_INVALID_PASSWORD

  logonPassword for this logonId is wrong.

- ECConstants.EC_PVC_LOGONSTATUS

  The command has been executed in logon status.

PVCBufferUrl

This command enables the buffering of input field data placed in multiple pages, and then sends the data to one command. It temporarily saves parameters of the destination URL in the WCS database PVCBUFFER table. Buffers that have not been updated for some time will be made inaccessible by specifying a bufferTimeout (set in minutes) in the configuration file.

When there is no specification of bufferTimeout, it will time out in five minutes. When 0 is specified, it will not time out. In addition to buffering the function of parameters, it enables the calling of different URLs for each button from forms that cannot use JavaScript.

Behavior

1. Checks parameters.
2. For b_new, it creates a new record inside the PVCBUFFER and displays a specified redirect view. For b_update or b_exec, it checks on buffertimeout using the value of `bufferTimeout` in the XML configuration file. If records in this table are not accessed for the period specified as `bufferTimeout, the` user cannot access the buffered parameters any more. In this case, it displays a specified error view.
3. In the case of b_update or b_exec, it adds or changes the parameter in PVCBUFFER.
4. In the case of b_update, it displays specified redirect view.
5. In the case of b_exec, it checks whether it is an excluded command or a password-locked command that is defined in the configuration file. If the command is an excluded command, it displays an error view but does not execute the command. In the case of a password-locked command, ReEnterPassword form is displayed, checks the password and executes the command. Then it starts up the command specified in the URL of PVCBUFFER. It receives a view of the command result and specifies it on the redirect view.

Commands executed internally

- Command specified in the URL in the case of b_new.

Tasks command executed internally

- None

Syntax

The syntax for the PVCBufferUrl command is shown in Figure B-4.

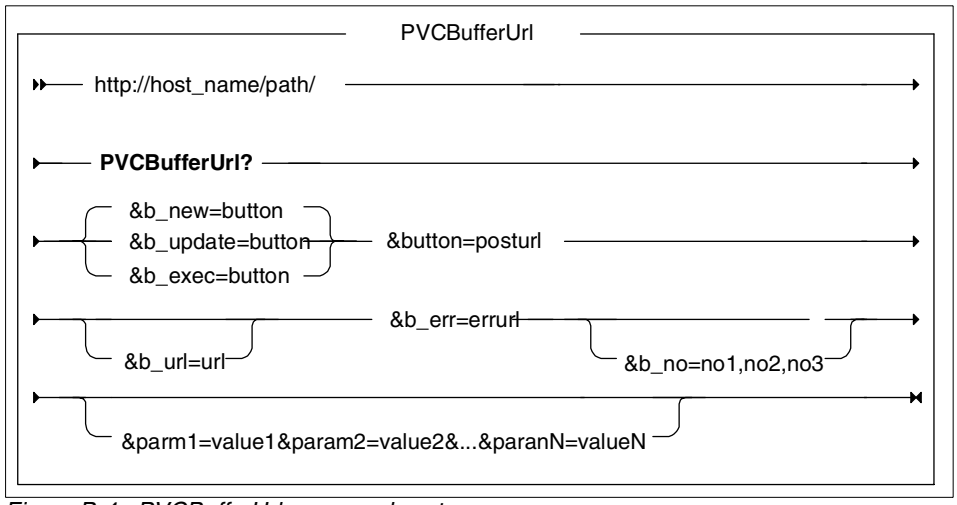


Figure B-4 PVCBufferUrl command syntax

Arguments

By choosing b_new, b_update or b_exec, some parameters become optional.

- http://host_name/path

  The fully qualified name of the server and the configuration path.

- b_new

  Creates a new parameter buffer.

- b_update

  Updates the present parameter.

- b_exec

  First it updates the present parameter. Next, it executes the targeted command with the parameter buffer.button_iSame value of b_new, b_update, and b_exec. The URL is called after handling the buffering.

- b_url

  The URL of the targeted command when executing b_exec.

- b_errurl

  The URL that is called when the PVCBufferUrl command finishes abnormally.

- b_no

  Parameters that are not to be buffered within the specified parameter.

Example

Creates a new buffer.

```
PVCBufferUrl?b_new=Next&Next=PVCBufferUrl2Form&b_err=PVCBufferUrlErrorView&
b_url=PVCRegistration&logonId=logon&b_no=b_new
```

Updates the contents of the buffer.

```
PVCBufferUrl?b_update=Next&Next=PVCBufferUrl3Form&b_err=PVCBufferUrlErrorVi
ew&p1=v1&p2=v2&p3=v3&p4=v4&b_no=p1,p3
```

Updates the contents of the buffer and executes the targeted command specified in b_new.

```
PVCBufferUrl?b_exec=Execute&b_err=PVCBufferUrlErrorView&p5=v5
```

Error views

- PVCBufferUrlErrorView

Error codes

- ECConstants.EC_PVC_B_URL

  b_url is not specified when specifying b_new.

- ECConstants.EC_PVC_B_ERR

  b_errurl is not specified.

- ECConstants.EC_PVC_BUF_ACTION

  Neither b_new, b_update, nor b_exec is specified.

- ECConstants.EC_PVC_BUF_TIMEOUT

  A buffer timeout occurred in b_update and b_exec.

- ECSecurityConstants.ERR_INVALID_USERTYPE

  The status is not logon when executing ReEnterPassword command.

- ECSecurityConstants.ERR_INVALID_PASSWORD

  The password is wrong in ReEnterPassword command.

ReEnterPassword

This command adds the given reenterpw parameter to the specified URL and redirects it. Usually the ReEnterPassword command is used by a JSP which is assigned to the ReEnterPassowrdForm. The ReEnterPasswordForm is called when executing the command without a password in password-locked status.

Behavior

1. Extracts `reenterpw` and `URL` parameter from the request.
2. Appends `reenterpw` to the value of the URL, used for location redirection.
3. Sends the redirection instructions to the client's device.

Commands executed internally

- None

Tasks command executed internally

- None

Syntax

The syntax for the ReEnterPassword command is shown in Figure B-5.

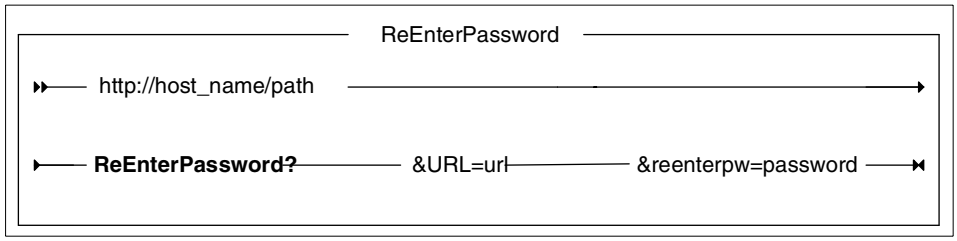


Figure B-5 ReEnterPassword command syntax

Arguments

- url

  The URL that needs password reentry

- password

  The password of a logged-on user

Example

The command adds a reenterpw parameter of OrderProcess, and executes it.

```
http://myhost/webapp/commerce/store/servlet/ReEnterPassword?reenterpw=passw
d&URL=OrderProcess
```

Error views

Generic error page is shown when required parameters are missing.

Error codes

- None

C

PvC data bean reference

The PvC data beans provide access to the WCS database for mobile device and user information. The PvC data beans are used within JSPs. For each of the data beans, we describe how to use it, provide the data available for the data bean and provide a sample of how to implement the data bean within a JSP intended to be used by a mobile device.

The appendix is organized into the following sections:

- PVCBufferDataBean
- UserPVCDeviceDataBean

PVCBufferDataBean

Using this data bean you can access parameters buffered by the PVCBufferUrl command from the WCS database. This data bean can be accessed only by registered users; otherwise this data bean is not populated.

How to use this data bean

To use this data bean, you don't need to pass parameters to it. You only have to invoke the DataBeanManager activate method, as shown in Example C-1.

Example: C-1 PVCBufferDataBean

```
PVCBufferDataBean buffer = new PVCBufferDataBean();
try {
   com.ibm.commerce.beans.DataBeanManager.activate (buffer, request);

} catch (Exception e) {
   strbuffer = "bad";
}
```

After the activate method has been invoked, data bean objects will have data corresponding to the record in the PVCBUFFER table that is being used by the current user. You can write code to access buffered parameters using the getParameter(String) method, which returns the value of the parameter. This methods allows you to access the value of the buffered parameters by name.

UserPVCDeviceDataBean

This data bean can retrieve information about the user device.

How to use this data bean

To use this data bean, you don't need to pass parameters to it. You only have to invoke the DataBeanManager activate method, as seen in Example C-2.

Example: C-2 DataBeanManager method

```
UserPVCDeviceDataBean pvcDeviceBean = new UserPVCDeviceDataBean();
try {
   com.ibm.commerce.beans.DataBeanManager.activate (pvcDeviceBean, request);
} catch (Exception e) {
   // Handle the exception here.
   // If no object is found exception will be thrown.
}
```

After the activate method has been invoked, data bean objects will have data corresponding to the user device that is being used for access. Information can be retrieved by using the methods as seen below.

Available methods

- public String getAddress1()

  Returns the value of ADDRESS1 in the USERPVCDEV table.

- public String getAddress1Type()

  Returns the value of ADDRESS1TYPE in the USERPVCDEV table.

- public String getAddress2()

  Returns the value of ADDRESS2 in the USERPVCDEV table.

- public String getAddress2Type()

  Returns the value of ADDRESS1 in the USERPVCDEV table.

- public String getDeviceFormatId()

  Returns the value of DEVICEFMT_ID in the USERPVCDEV table.

- public Integer getDeviceFormatIdInEJBType()

  Returns the device format ID in the Integer type.

- public String getDeviceIdentifier()

  Returns DEVICEIDENTIFIER in the USERPVCDEV table. This data is the same as the value of TERMINAL in the PVCSESSION table. This value is the subscriber ID of the client's device.

- public String getPreferredTimeout()

  Returns the value of PREFERREDTIMEOUT in the USERPVCDEV table. This value is copied from `preferredTimeout` in the XML configuration file and used calculate logon timeout.

- public String getProtect()

  Returns the value of PROTECT in the USERPVCDEV table. If the value is 1, it means that the client device needs to reenter the password to execute the password-locked commands. Password-locked commands can be specified in the XML configuration file.

- public Integer getProtectInEJBType()

  Returns the value of PROTECT in the Integer type.

- public String getTransportId()

  Returns the value of TRANSPORT_ID in the USERPVCDEV table.

- public Integer getTransportIdInEJBType()

Returns the value of TRANSPORT_ID in the Integer type.

- public String getUserId()

  Returns the value of USERS_ID in USERPVCDEV table. This value is the same as the ID of the user who is accessing the WCS server.

- public Long getUserIdInEJBType()

  Returns the value of USERS_ID in the Long type.

- public String getUserPVCDeviceId()

  Returns the value of USERPVCDEV_ID in the USERPVCDEV table. This value is the primary key of the record in USERPVCDEV table. Value is corresponding to the key for the client's device that is being used for the access.

- public Long getUserPVCDeviceIdInEJBType()

  Returns the value of USERPVCDEV_ID in the Long type.

D

PvC database tables and content management reference

The PvC database tables are used for content management, session control and to store information about the user and device. This appendix provides a description of each of the PvC database tables.

WebSphere Commerce Suite V5.1 provides the following PvC database tables:

- PVCDEVMDL
- PVCMDLSPEC
- PVCDEVSPEC
- PVCBinding
- PVCSession
- PVCBuffer
- Content management reference

PVCDEVMDL

This table stores model information for mobile devices and is used for content management.

Each record in the PVCDEVMDL table corresponds to existing models of PvC devices. A corresponding record of a certain model can be found in the table, by name of the adapter and model name returned by the adapter. The server uses this table to search for a suitable record in the PVCMDLSPEC table for the device accessing the server. If a server finds a non-existing model, it will automatically insert a new record into this table. By checking the contents of this table, you can determine what kind of devices are accessing your Web site.

A blank model name has special meanings in this table. The relationship between a blank model name record and a record in the PVCMDLSPEC table becomes a common relationship for all models managed by the same adapter which do not have specific model-spec mapping in the PVCMDLSPEC table.

If you want to categorize models into groups and want to prepare different JSPs for each group, you need to add records for the target devices. Alternatively, by accessing the WCS server using the mobile device, a new record will be added to this table if it does not exist. To define new model information, you need to fill the columns displayed in the PVCDEVMDL table.

For a working example of the configuration changes for content management, refer to 12.1, “Content management configuration” on page 226.

Table D-1 PVCDEVMDL table

MODEL_ID	Reference number of this record. This must be unique. We recommend you use a negative number when you manually add new records to this table. Otherwise, when you modify this table manually, you will need to stop the WCS server. After you have finished work on this table, you need to update the value of COUNTER for the PVCDEVMDL table in the KEYS table to have a value larger than the maximum number of MODEL_ID in the PVCDEVMDL table.
MODELNAME	Name of model. Specifies the model name which the adapter returns.
SESSIONTYPE	Name of the adapter which returns the model name in the MODELNAME column.

VENDOR	Name of the vendor of the model. This column is prepared for your management use. The WCS server does not use this column at all. You can specify any desired value. This column allows a NULL value.
DESCRIPTION	Description of this record. This column is prepared for management use. WCS server does not use this column at all. You can specify any desired value. This column allows a NULL value.

PVCMDLSPEC

This table stores specification information and a directory of the content. One record can be shared by more than two devices.

This table shows specifications of devices and the location of the JSP root for these devices. The WCS server changes the contents to be sent to the client's device by selecting a suitable record for the device from this table. The WCS server not only selects suitable contents for the client device, but also sets attributes to the JSP. Content switching is enabled by the value of the CONTENTTYPE directory. If you want to prepare some types of JSP for the same view, you need to create a record for each type. Or, if you want to set various data to a JSP file which depends on the capability of the client, you also need to create a record for each type.

Table D-2 PVCMDLSPEC table

SPEC_ID	Reference number of this record. This must be unique.
SPECNAME	Name of the definition. This column is prepared for management use. The WCS server does not use this column at all. You can specify any desired value.
SESSIONTYPE	Name of the adapter which primarily uses this configuration.
MAXCONTENTLENGTH	Maximum length of content which the browser can receive. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.

MAXURLLENGTH	Maximum length of the URL which the browser can access. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.
LCDWIDTH	Width of the LCD panel on the device. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.
LCDHEIGHT	Height of the LCD panel on the device. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.
LCDCOLORS	Number of colors that the LCD panel on the device can display. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.
LCDMONOCHROME	Indicates whether the LCD panel on the device is monochrome or not. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.
IMAGEFORMAT	Available format for image. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.
SOUNDFORMAT	Available format for sound. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.
DOCUMENTFORMAT	Available format for document. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.

DOCUMENTVERSION	Available version of the document format. You can specify any desired value for this column to use this data from your JSP file. The value of this field is accessible from the JSP.
CONTENTDIR	Name of the directory which contains contents for the device. The value of this column will be used by the content switching function and the value of the column will be inserted into the JSP file search path when the view file is processed. The value of this field is accessible from the JSP.
DESCRIPTION	Description of this record. This column is prepared for your management use. The WCS server does not use this column at all. You can specify any desired value. This column allows a NULL value.

PVCDEVSPEC

This table stores the relationship between model_id and spec_id. By using this table, device models which have similar specifications can be categorized to one record in the PVCDEVSPEC table.

In addition to the tables used for content management configuration, WCS V5.1 includes the following PvC database tables used for session management and URL buffering. These tables do not need to be configured.

Table D-3 PVCDEVSPEC table

MDLSPC_ID	Reference number of this record. This must be unique.
STOREENT_ID	Reference number of the store. Setting the value to 0, is used when the relationship is site wide. Other values mean that the relationship is unique to the store.
MODEL_ID	Reference number of the model.
SPEC_ID	Reference number of the spec.

PVCBinding

This table is used for session management and does not need to be configured.

PVCSession

This table is used for session management and does not need to be configured.

PVCBuffer

This table is used for stored buffered data used by the PVCBufferUrl command.

Content management reference

This section provides reference information necessary for content management configuration.

PVCDEVMLD table: define adapter default model

We need to configure the necessary settings for the PVCDEVMDL table for all devices whose sessions are managed by the PvC adapter.

- We define all mobile devices which are detected of specific browser to the content directory of JSP files separate from PC browser client JSPs.
- To represent adapter common settings, we need to prepare one record which has a blank MODELNAME.
- The SESSIONTYPE of the record should be the same as the adapter name. From now on, we will call the record which has a blank MODELNAME and SESSIONTYPE the adapter default model.
- When a model which is not listed in PVCDEVMDL is found, the server will automatically insert a new record into PVCDEVMDL.
- If there is no specific model-spec mapping in the PVCMDLSPEC table, the adapter default model will be used to search for a suitable record in PVCDEVSPEC table.

Example D-1 displays SQL sample for the adapter default model for the UPLinkGatewayAdapter to be created in the PVCDEVMDL table.

Example: D-1 Sample - sql to update the PVCDEVMDL table

```
insert into PVCDEVMDL (
   MODEL_ID,
   MODELNAME,
   SESSIONTYPE,
   VENDOR,
   DESCRIPTION
) values(
   1,
   '',
   'UPG',
   'Default',
   'Default model for UPLinkGatewayAdapter'
);
```

PVCDEVSPEC table: define minimum spec and JSP root

By configuring the PVCDEVSPEC table, we define the JSP root directory and minimum spec information for the PvC adapter mobile devices.

- The CONTENTDIR is the key column in this table.
- The server changes the search path of JSPs which will be processed as a result of command execution by the value in this column.
- From the JSP file, the value of the selected record in the PVCDEVSPEC table is accessible through an object of DeviceInfo class which is set as an attribute of the HTTP request for the JSP file.

Note: Before you run this script on your system, check to see if you already have records in this table.

- Replace the value of SPEC_ID with a value which is not a duplicate of an existing record.
- Specify the adapter name in the SESSIONTYPE column.
- For other fields, you can set whatever you want. These values will be accessible through the DeviceInfo object.

Example D-2 provides a sample of the SQL statements required to insert the adapter default setting into the PVCDEVSPEC table. For example, the sample sets UPG for CONTENTDIR.

Example: D-2 Sample - sql to update PVCDEVSPEC table

```
insert into PVCDEVSPEC (
   SPEC_ID,
   SPECNAME,
```

```
   SESSIONTYPE,
   MAXCONTENTLENGTH,
   MAXURLLENGTH,
   LCDWIDTH,
   LCDHEIGHT,
   LCDCOLORS,
   LCDMONOCHROME,
   IMAGEFORMAT,
   SOUNDFORMAT,
   DOCUMENTFORMAT,
   DOCUMENTVERSION,
   CONTENTDIR,
   DESCRIPTION
) VALUES (
   1,
   'UPG default',
   'UPG',
   4096,
   256,
   120,
   80,
   2,
   '1',
   'bmp',
   '',
   'HDML',
   '1.0',
   'UPG',
   'minumum spec'
);
```

PVCMDLSPEC table: PVCDEVMDL and PVCDEVSPEC relationship

This table defines the relationship between PVCDEVMDL and PVCDEVSPEC. We set up this relationship in order for all devices which come from the adapter to share the same setting for the content JSP directory and other information in the PVCDEVSPEC table.

Example D-3 provides a sample of the SQL required to update the PVCMDLSPEC table to define the relationship between the adapter default model and the record in PVCDEVSPEC table.

Example: D-3 Sample - sql to update the PVCMDLSPEC table

```
insert into PVCMDLSPEC (
   MDLSPC_ID,
   STOREENT_ID,
   MODEL_ID,
```

```
   SPEC_ID
) values (
   1,
   0,
   1,
   1
);
```

Once these records are inserted into the tables, JSPs used for connections which are handled by the adapter whose name is UPG will be picked up from the UPG directory under the actual document root directory.

E

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

```
ftp://www.redbooks.ibm.com/redbooks/SG246171
```

Alternatively, you can go to the IBM Redbooks Web site at:

```
ibm.com/redbooks
```

Select the **Additional materials** and open the directory that corresponds to the redbook form number, SG246171.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
SG246171.zip	PvC Fashion sample store and source code samples

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 10 MB minimum
Operating System: Windows NT or Windows 2000
Processor: 733 MHz or higher
Memory: 512 MB or higher

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder. The contents of the unzipped file are used in many of the examples throughout this redbook.

Refer to Chapter 9, “m-commerce sample store and sample code” on page 179 for instructions on setting up the sample store.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications see “How to get IBM Redbooks” on page 373.

- *WebSphere Commerce Suite V5.1 Handbook*, SG24-6167 redpiece available at:

  `http://www.ibm.com/redbooks` (expected redbook publish date August 2001)

- *WebSphere Commerce Suite V5.1 Customization and Transition Guide*, SG24-6174 redpiece available at:

  `http://www.ibm.com/redbooks` (expected redbook publish date July 2001)

- *WebSphere V3.5 Handbook*, SG24-6161
- *e-Commerce Patterns using WebSphere Commerce Suite, Patterns for e-business Series*, SG24-6156
- *Patterns for e-business: User-to-Business Patterns for Topology 1 and 2 using WebSphere Advanced Edition*, SG24-5864
- *An Introduction to IBM Everyplace Suite Version 1.1, Accessing Web and Enterprise Applications*, SG24-5995
- *IBM WebSphere Transcoding Publisher V1.1, Extending Web Applications to the Pervasive World*, SG24-5965

Other resources

These publications are also relevant as further information sources:

- *Programmer’s Guide, IBM WebSphere Commerce Suite V5.1*, product guide found on WebSphere Commerce Suite V5.1 CD.
- *Fundamentals, IBM WebSphere Commerce Suite V5.1*, product guide found on WebSphere Commerce Suite V5.1 CD.
- Arehart, Charles, et al, *Professional WAP*, Wrox, July 2000, ISBN 1861004044

- Dornan, Andy, *The Essential Guide to Wireless Communications Applications: From Cellular Systems to WAP and M-Commerce*, Prentice Hall PTR, November 2000, ISBN 0130317160
- *Nokia WAP Toolkit 2.1 User's Guide*, found at:

  `http://www.forum.nokia.com/wapforum/main/1,6668,1_1_50_20,00.html`
- *Nokia WAP Toolkit 2.1 Developer's Guide*, found at:

  `http://www.forum.nokia.com/wapforum/main/1,6668,1_1_50_20,00.html`
- *UP.SDK 4.1 Developer's Guide*, product documentation is provided with the software developer's kit (SDK) found at:

  `http://developer.phone.com/download/index.html#sdk`
- *UP.SDK 3.2 for WML Developer's Guide*, product documentation is provided with the software developer's kit (SDK) found at:

  `http://developer.phone.com/download/index.html#sdk`
- *Palm OS Companion and Reference*, found at:

  `http://www.palmos.com/dev/tech/docs/`
- *Palm OS Development Tools Guide*, found at:

  `http://www.palmos.com/dev/tech/docs/`

Referenced Web sites

These Web sites are also relevant as further information sources:

- `http://http://www.forum.nokia.com/main/1,,1_1,00.html/` Nokia WAP developers forum.
- `http://developer.phone.com/` Openwave developer program.
- `http://www.palmos.com/dev/tech/tools/` Palm OS development tools.
- `http://www.palmos.com/dev/tech/docs/` Palm OS documentation.
- `http://www.nttdocomo.com/i/` NTT DoCoMo official site for i-mode.
- `http://www.tca.or.jp/index-e.html/` Total number of subscribers for service providers in Japan.
- `http://www.au.kddi.com/ezweb/contents/index.html/` KDDI eZweb home page, service provider in Japan.
- `http://www.j-sky.j-phone.com/` J-Sky home page, J-phone service in Japan.
- `http://www.umts-forum.org/` UMTS forum.

- http://java.sun.com/j2me/ General information on Java 2 Micro Edition (J2ME).
- http://www.vxml.org/ General information on voice xml.
- http://www.ibm.com/pvc/tech/library.shtml/ IBM Pervasive Computing (PvC) technical library.
- http://www.ibm.com/pvc/products/mobile_connect/index.shtml/ IBM Mobile Connect home page.
- http://www.ibm.com/software/data/db2/everyplace/ DB2 Everyplace home page.
- http://www.ibm.com/software/ts/mqseries/everyplace/ MQSeries Everyplace home page.
- http://developer.phone.com/resources/markup.html/ General information about selecting markup languages.

How to get IBM Redbooks

Search for additional Redbooks or redpieces, view, download, or order hardcopy from the Redbooks Web Site

ibm.com/redbooks

Also download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web Site for information about all the CD-ROMs offered, updates and formats.

Special notices

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.
Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of other companies:

Glossary

AMPS. Advanced Mobile Phone Services. A common analog cellular telephone service standard.

Applet. A Java applet is a small application program that is downloaded to and executed on a Web browser or network computer. A Java applet typically performs the type of operations that client code would perform in a client/server architecture. It edits input, controls the screen, and communicates transactions to a server, which in turn performs the data or database operations.

API. Application Program Interface.

Bean Managed Persistence. Bean Managed Persistence (BMP) is a term used to describe a type of entity Enterprise JavaBean where the bean developer specifies how the bean is to be persisted to a database by writing Java code in the appropriate methods to perform the tasks required.

Bluetooth. A short range (10 to 100 m.) wireless radio transport.

Cache. A cache stores cachable responses in order to reduce the response time and network bandwidth consumption on future, equivalent requests. Any client or server may include a cache, though a cache cannot be used by a server while it is acting as a tunnel.

Cache server. Some networks use a cache server to store Web pages and other data, so that if the same pages are requested frequently, they can be served from the cache rather than repeatedly retrieved from external Web servers. The external cache is an HTTP proxy such as IBM Web Traffic Express. IBM WebSphere Transcoding Publisher can use it to store and retrieve transcoded Web pages and intermediate results to avoid repeating the transcoding of frequently accessed pages, delivering better performance.

CDMA. Code Division Multiple Access. A second generation digital cellular network standard.

CDPD. Cellular Digital Packet Data. Designed to work as an overlay on analog cellular networks.

CGI. Common Gateway Interface. A standard way of communicating between different processes.

Cell phone. CELLular telePHONE is the first ubiquitous wireless telephone. Originally analog, all new cellular systems are digital. This has enabled the cell phone to turn into a smart phone that has access to the Internet.

Clustering. Clustering is a technique used to provide scalability through the use of multiple copies of an application on the same machine or on separate machines. Careful management of the different applications is necessary to ensure that they work together effectively. WebSphere has limited clustering support in Version 2.x and more support in Version 3.0.

cHTML. Compact HTML is a more efficient variation of HTML specifically designed for use by the i-mode wireless service.

Container Managed Persistence. Container Managed Persistence (CMP) is a term used to describe a type of entity Enterprise JavaBean where the code to persist the bean to a database is generated at deployment time by the EJB container.

ESS. Enterprise Solution Structure defines a set of technical reference architectures that are included in SIMethod.

GSM. Global System for Mobile Communications is a digital cellular phone technology based on TDMA that is the predominant system in Europe, but is also used around the world. Developed in the 1980s, GSM was first deployed in seven

European countries in 1992. Operating in the 900 MHz and 1.8 GHz bands in Europe and the 1.9 GHz PCS band in the U.S., GSM defines the entire cellular system, not just the air interface (TDMA, CDMA, etc.). As of 2000, there were more than 250 million GSM users, which is more than half of the world's mobile phone population.

e-business. e-business is a term used by IBM to describe the use of Internet technologies to transform business processes. What this means in practice is using Internet clients such as Web browsers as front ends for applications that access back-end legacy systems to allow greater access. See `http://www.software.ibm.com/ebusiness` for more information.

Enterprise Java Beans. Despite the name, Enterprise Java Beans (EJBs) are not Java Beans. Enterprise Java Beans are server-side Java components that are designed for distributed environments. They do not exist in isolation but rather are deployed in containers that provide services such as security, naming and directory services, and persistent storage. WebSphere Application Server is just such a container. See `http://java.sun.com/products/ejb/` for more information.

EPOC. A 32-bit operating system for handheld devices from Symbian Ltd. Used in Psion and other handheld computers, it supports Java applications, e-mail, fax, infrared exchange, data synchronization with PCs and includes a suite of PIM and productivity applications. See `http://(www.symbian.com` for more information.

Gateway. A server which acts as an intermediary for some other server. Unlike a proxy, a gateway receives requests as if it were the origin server for the requested resource; the requesting client may not be aware that it is communicating with a gateway. Gateways are often used as server-side portals through network firewalls and as protocol translators for access to resources stored on non-HTTP systems.

GSM. Global System for Mobile Communications. Widely used in Europe.

GPRS. General Packet Radio Service or GPRS is an enhancement to the GSM mobile communications system that supports data packets. GPRS enables continuous flow of IP data packets over the system for such applications as Web browsing and file transfer. GPRS differs from GSM's short messaging service (GSM-SMS) which is limited to messages of 160 bytes in length.

HTML. Hyper Text Markup Language is a document format used on the World Wide Web. Web pages are built with HTML tags, or codes, embedded in the text. HTML defines the page layout, fonts and graphic elements as well as the hypertext links to other documents on the Web. Each link contains the URL, or address, of a Web page residing on the same server or any server worldwide, hence the term “World Wide” Web.

HDML. Handheld Device Markup Language is a specialized version of HTML designed to enable wireless pagers, cell phones, mobile phones and other handheld devices to obtain information from Web pages. HDML was developed by Phone.com (formerly Unwired Planet) before the WAP specification was standardized. It is a subset of WAP with some features that were not included in WAP. AT&T Wireless launched the first HDML-based service in 1996.

HTTP proxy. An HTTP proxy is a program that acts as an intermediary between a client and a server. It receives requests from clients, and forwards those requests to the intended servers. The responses pass back through it in the same way. Thus, a proxy has functions of both a client and a server. Proxies are commonly used in firewalls, caching and transcoding machines.

IBM WebSphere Transcoding Publisher. IBM WebSphere Transcoding Publisher is network software that modifies content presented to users based on the information associated with the request, such as device constraints, network constraints, user preferences, and organizational policies. Transforming content can reduce or eliminate the need to maintain multiple versions of data or applications for different device types and network service levels.

Image Transcoder. Image Transcoder is a transcoder that can scale, modify quality, and

modify color levels in JPEG and GIF images. Additionally, the Image Transcoder can convert JPEGs to GIFs for devices that do not render JPEGs.

i-mode. A packet-based information service for mobile phones from NTT DoCoMo (Japan). i-mode provides Web browsing, e-mail, a calendar, chat rooms, games, and customized news. It was the first smart phone system for Web browsing and its popularity grew very quickly after its introduction in 1999. i-mode is a proprietary system that uses a subset of HTML, known as cHTML, in contrast to the global WAP standard that uses a variation of HTML, known as WML. The i-mode transfer rate is 9600 bps, but is expected to increase to 384 kbps in 2001, using W-CDMA.

IrDA. The Infrared Data Association develops standards for wireless, infrared transmission systems between computers. With IrDA ports, a laptop or PDA can exchange data with a desktop computer or use a printer without a cable connection. IrDA requires line-of-sight transmission like a TV remote control. IrDA products began to appear in 1995. See `http://www.irda.org` for more information.

JavaBeans. JavaBeans are Java components designed to be used on client systems. They are Java classes that conform to certain coding standards. They can be described in terms of their properties, methods and events. JavaBeans may be packaged with a special descriptor class called a BeanInfo class and special property editor classes in a JAR file. Java Beans may or may not be visual components. See `http://www.javasoft.com/beans/docs` for more information.

JavaServer Pages (JSP) . JSPs provide a simplified, fast way to create dynamic Web content. JSP technology enables rapid development of Web-based applications that are server and platform independent. JavaServer Pages are compiled into servlets before deployment.

JDBC. JDBC is a Java API that allows Java programs to communicate with different database management systems in a platform-independent manner. Database vendors provide JDBC drivers with their platforms that implement the API for their database, allowing the Java developer to write applications to a consistent API no matter which database is used.

JNDI. Java Naming and Directory Interface (JNDI) is an API that allows Java programs to interface and query naming and directory services in order to find information about network resources. JNDI is used in WebSphere to provide a directory of Enterprise Java Beans. See `http://java.sun.com/products/jndi/index.html` for more information.

JSP. See JavaServer Pages.

m-commerce. Mobile commerce refers to the use of mobile devices to partially or completely perform a transaction electronically from a commerce Web site for the exchange of goods or services for monetary consideration. Simply put, m-commerce is electronic commerce using a mobile device such as a mobile phone or PDA.

Mobile device. A mobile device is a portable, generally small, wireless device that can be used to access the Internet via a browser. It includes a wide range of capability and functionality. Mobile devices include mobile phones, wireless PDAs, and wireless laptops.

Mobile phone. A mobile phone is a wireless smart phone that has a microbrowser to access Internet content. Other names for a mobile phone include cell phone and wireless phone.

MQe. See MQ Everyplace.

MQ Everyplace. It is designed to satisfy the messaging needs of lightweight devices and the requirements that arise from the use of fragile communication networks.

PDA. Personal Digital Assistant.

PCS. Personal Communications Services (PCS) are wireless services that emerged after the U.S. Government auctioned commercial licenses in 1994 and 1995. This radio spectrum in the 1.8-2 GHz range is typically used for digital cellular transmission that competes with analog and digital services in the 800 MHz and 900 MHz bands.

Persistence. Persistence is a term used to describe the storage of objects in a database to allow them to persist over time rather than being destroyed when the application containing them terminates. Enterprise Java Bean containers such as WebSphere provide persistence services for EJBs deployed within them.

PKI. Public Key Infrastructure.

PvC. Popular short form within IBM for pervasive computing.

Proxy. Transcoding Publisher connects through a proxy server that is configured with a firewall to manage network traffic and to protect your network from outside intrusion.

Push. Push refers to a technology that sends data to a program without the program's request (unsolicited).

RMI. Remote Method Invocation (RMI) is a lightweight distributed object protocol that allows Java objects to call each other across a network. RMI is part of the core Java specification. See `http://java.sun.com/products/jdk/rmi/index.html` for more information.

Scalability. Scalability is an abstract attribute of software that refers to its ability to handle increased data throughput without modification. WebSphere handles scalability by allowing execution on a variety of hardware platforms that allow increased performance and clustering.

Servlets. Servlets are Java classes that run on Web servers to provide dynamic HTML content to clients. The servlets take as input the HTTP request from the client and output dynamically generated HTML. For more information, see `http://www.software.ibm.com/ebusiness/pm.html#Servlets`.

SMS. Short Message Service or SMS is text message service that enables short messages of generally no more than 140-160 characters in length to be sent and transmitted from a cellphone. SMS is supported by GSM and other mobile communications systems. Unlike paging, short messages are stored and forwarded in SMS centers.

SOCKS. A SOCKS server is a proxy server that uses a special protocol, sockets, to forward requests. Transcoding Publisher connects through a SOCKS server that is configured with a firewall to manage network traffic and to protect your network from outside intrusion (it supports Versions 4 and 5 SOCKS servers).

SSL. Secure Sockets Layer. A secure protocol used for authentication and encryption. SSL can be used over HTTP, RMI, Telnet and other protocols.

Stand-alone Network Proxy. When the IBM WebSphere Transcoding Publisher is used as a normal proxy in a browser, the data that flows from the original source will be transcoded in the proxy according to the device and network profile needed.

Stylesheet Transcoder. Stylesheet Transcoder is a transcoder that selects the style sheet and applies it to an input Extensible Markup Language (XML) document to produce a version that is appropriate for the target device.

TCP/IP. TCP/IP is a set of protocols developed to allow cooperating computers to share resources across a network.

TDMA. Time Division Multiple Access. A second-generation digital cellular network standard.

Text Transcoder. Text Transcoder is a transcoder that can modify elements of a text document based on device, network and, potentially, user preference information. The primary use of this Text Transcoder is to modify Hypertext Markup Language (HTML) documents to remove unsupported elements, reduce space usage, replace features such as images or frames with links, and otherwise tailor documents to allow for their better display on devices with screen limitations.

TLS. Transport Layer Security. The standard (IEFT) security protocol on the Internet. It is expected to eventually supersede SSL.

Transcoder. Transcoder is a program that modifies the content of a document.

Transcoding. Transcoding is a new technology that gives you the ability to make Web-based information available on handheld and other new type devices economically and efficiently, or on the slow network connections like a dial up modem connection. With transcoding, users receive information (text and images) tailored to the capabilities of the devices they are using and also tailored to the capacity of the network being used.

Transcoding is also the process whereby the MEGs modify the request and generate the original resource and all of the document (or resource) editing (or transcoding).

Tunnel. A tunnel is an intermediary program which acts as a blind relay between two connections.

UMTS. Universal Mobile Telecommunications System is the European implementation of the 3G wireless phone system. UMTS, which is part of IMT-2000, provides service in the 2 GHz band and offers global roaming and personalized features. Designed as an evolutionary system for GSM network operators, multimedia data rates up to 2 Mbps are expected using the W-CDMA technology. In the meantime, GPRS and EDGE are interim steps that will speed up wireless data for GSM. For more information, visit http://www.umts-forum.org.

URL. Uniform Resource Locator. The URL specifies the Internet address of a file stored on a host computer connected to the Internet.

VXML. Voice XML is an extension of XML that defines voice segments and enables access to the Internet via telephones and other voice-activated devices. AT&T, Lucent and Motorola created the Voice XML Forum to support this development. For more information, visit http://www.vxml.org.

Voice XML. See VXML.

WAP. Wireless Application Protocol. The point of this standard is to serve Internet contents and Internet services to wireless clients and WAP devices, such as mobile phones and terminals. The authoritative source for WAP is http://www.wapforum.org.

WAS. IBM WebSphere Application Server.

Web Application Servers. A Web application server is a software program designed to manage applications at the second tier of three-tier computing, that is, the business logic components. A Web application server manages applications that use data from back-end systems, such as databases and transaction systems, and provides output to a Web browser on a client. For more information see http://www.software.ibm.com/ebusiness/appsrvsw.html

Web browser. To access the World Wide Web, you must use a Web browser. A browser is a software program that allows users to access and navigate the World Wide Web.

Wireless network. Used to transmit data between wireless devices such as a mobile phone, PDA, or personal computer without the use of a physical cable or wire.

Wireless service provider. An organization that provides wireless services, including cellular services, satellite services and ISPs.

Wireless LAN. A wireless LAN is a local area network that transmits over the air, typically in an unlicensed frequency such as the 2.4 GHz band. A wireless LAN does not require lining up devices for line of sight transmission, as IrDA does. Wireless access points (base stations) are connected to an Ethernet hub or server and transmit a radio frequency over an area of several hundred to a 1000 feet, which can penetrate walls and other non-metal barriers. Roaming users can be handed off from one access point to another like a cellular phone system. Laptops use wireless modems that plug into an existing Ethernet port or that are self contained on PC cards, while stand-alone desktops and servers use plug-in cards (ISA, PCI, etc.).

WLP. A modified version of the Point-to-Point Protocol (PPP) used by the IBM Wireless

Gateway to support wireless (non-WAP) client devices.

WML. Wireless Markup Language. XML-based, WML tags are used to mark up content in decks for WAP-enabled devices.

WTE. Web Traffic Express. An IBM caching proxy.

WTLS. Wireless Transport Layer Security. A simplified version of TLS designed specifically for WAP devices. It uses mini-certificates.

WTP. WebSphere Transcoding Publisher.

WWW. The World Wide Web (known as the Web) is a system of Internet servers that supports hypertext to access several Internet protocols on a single interface.

X.509. A digital certificate specification used by SSL and TLS. Mini-certificates are used by WTLS.

XML. XML, or Extensible Markup Language, is a platform-independent and application-independent way of describing data using tags. XML (a subset of SGML) is similar to HTML in that it uses tags to describe document elements, but different in that the tags describe the structure of the data rather than how the data is to be presented to a client. XML has the ability to allow data providers to define new tags as needed to better describe the data domain being represented. For more information see `http://www.software.ibm.com/xml`.

XSL. Extensible Style Language. XSL stylesheets are documents that describe a mapping between XML documents and visual data that can be presented to a client in a browser or mini-browser.

Index

Numerics

3G 39
  IMT-2000 39
4thpass Kbrowser 60
503i 49

A

Advanced Mobile Phone System
  See AMPS
AMPS 20, 38
AvantGo AvantGo 59

B

B2B 5
B2C 5
Bluetooth 69

C

Casio E-125 58
cdma2000 40
cdmaOne 40
CDPD 38
  network architecture 38
Cells 21
Cellular Digital Packet Data
  See CDPD
checkDeviceFormat 88, 213
  creating the method 215
  description 213
cHTML 66
Circuit-switched networks 22
Command registration
  WCS command 263
Compact HTML
  See cHTML
Compaq iPac 58
Content 64
  auto selection 242
  deployment 238
  managed by adapter 242
  managed by model 242
Content development 233
  advanced example 238
Content management 226
  auto content selection 229
  configuration 89, 231
  define the adapter default model 228
  deploying multiple adapters 222
  description 227
  PvC database tables 226
  reference information 359
Cookies 73
  non-secure session cookie 73
  secure authentication cookie 73
Creating
  a PvC adapter 205
  custom PvC command 255
  device-specific content JSPs 225
CyberCash 137

D

DB2 Everyplace 126
Deploy sample store 183
Deployment
  content 238
  deploy the PvC adapter 88
  deploy the PvC adapter JAR 88
  PvC custom command 262
Development environment 160
  high level install steps 162
Device control 86, 94
  browser 95
  markup language 94
  mobile devices 94
Device detection 208
Device registration form 235
Device-specific content JSPs 232
Divided large form 236
Download the additional materials 180
Dual-Slot phones 132

E

EdgeMatrix WAPman 60
EPOC 62

Ericsson R380 58
Nokia 9210 Communicator 58
Ericcson
R380 273
Ericsson 40
R380 58
Everyplace Wireless Gateway 123
architecture 124
install 172
m-commerce considerations 125
NAS/RAS 170
overview 124
EWG
See Everyplace Wireless Gateway

G

General Packet Radio Service
See GPRS
getDeviceMode 88
getDeviceModel 216
creating the method 216
description 216
getTerminalId 88, 217
description 217
naming the adapter 218
GPRS 32
border gateway (BG) 35
class of device
class A 35
class B 35
class C 35
gateway GPRS support nodes (GGSN) 35
network infrastructure 34
packet control unit (PCU) 34
serving GPRS support nodes (SGSN) 34
GSM 24
architecture 28
authentication center (AuC) 30
authentication protocol 30
base station controller (BSC) 29
base transceiver station (BTS) 29
encryption key 31
equipment identity register (EIR) 30
gateway MSC 29
home location register (HLR) 29
mobile switching center (MSC) 29
network management center (NMC) 30
operations management center (OMC) 30
SIM 25
subscriber 27
visitor location register (VLR) 29

H

Handheld Device Markup Language
See HDML
Handover 21
Handspring Blazer 59
Handspring Visor 58
HDML 65
content management script 269
sample code 269
HDML toolkits 266
HP Jornada 58
HTML 64
HTTP protocol 41
HyperText Markup Language
See HTML

I

IBM WorkPad PC 58
IMEI 27
i-mode 46
implementation guidelines 291
NTT DoCoMo 46
IMT-2000 39
3G 39
Intellisync Browse-it 59
International Mobile Equipment Identifier
See IMEI
International Mobile Telecommunications-2000
See IMT-2000
ISP 67

J

J2ME 67
NTT DoCoMo 503i cell phone 49
JavaBean 120
J-PHONE 46

K

KDDI 46

L

LDAP 341
Logon timeout 97

M

Market study 74
Markup languages 64, 94
  cHTML 66
  HDML 65
  HTML 64
  WML 65
  XHTML 67
  XML 66
m-commerce
  application design 81
  application development 81
  considerations for WTP 122
  defined 5
  device control 86
  EWG considerations 125
  logon timeout security 97
  market study 74
  new business models 11
  objectives 5
  opportunities 7
  restricted command execution 98
  security enhancements 87
  selection guidelines 77
  session control 86
  testing the application 83
  URL buffering 87, 101
  WES considerations 112
m-commerce direct 76
  computing flow 76
  defined 76
  design guidelines 200
  development for WAP 275
  development process 203
  pros and cons 79
m-commerce using WTP 76
  application runtime 77
  content management script 306
  defined 76
  design guidelines 302
  pros and cons 80
  sample code 306
  sample HTML and XML JSPs 306
  sample PvC adapter 306
MIME filter 120, 123
Mobile Connect 127
Mobile devices 6, 57
  cons 63
  display color, size, res. 96
  pros 63
  wireless laptop 62
  wireless PDA 58
Mobile market penetration 8
  Europe 10
  Japan 8
  US 11
Mobile phone 6, 57
Mobile wallet 133
Mobitex 36
  external access protocols 37
  network architecture 36
MQSeries Everyplace (MQe) 125

N

NAS 170
Network proxy 119
Nokia 40
Nokia 9210 Communicator 58
Nokia WAP Toolkit V2.1 169
Nokia Wap Toolkit V2.1 272
NTT DoCoMo 40, 46
  503i 49
  503i Java architecture 50
  i-mode 46

O

Official site 47
OmniSky OmniSky 59

P

Packet-switched networks 22
Palm 58
  HTTP browsers 59
    AvantGo AvantGo 59
    Handspring Blazer 59
    Intellisync Browse-it 59
    OmniSky OmniSky 59
    Qualcomm EudoraWeb 60
  Palm Query Application 288
  Palm V 58
  Palm VII 58
  WAP browsers
    4thpass Kbrowser 60
    EdgeMatrix WAPman 60
  Web Clipping 284
    browser 60

Web clipping
design guidelines 284
development guidelines 286
example 288
testing 286, 287
Password reentry 234
Password reentry form 234
Payment technologies
dual-slot 132
mobile wallet 133
SET 130
SIM 134
SmartCards 130
SSL 130
WAP WTLS 131
PDA
Compaq iPac Pocket PC 58
EPOC 62
Handspring Visor 58
IBM WorkPad PC 58
Palm VII 58
PDC 39
Personal Digital Assistant
See PDA
Personal Digital Cellular
See PDC
PocketPC 61
Casio E-125 58
Compac iPac 58
HTTP browser 61
WAP browser 61
PQA
See Palm Query Application
PrintHttpRequest 208
Proxy 123
Psion Series 58
PvC adapter
checkDeviceFormat method 213
create an adapter class 210
creating a PvC adapter 206
definition 89
definition XML file 220
deploying multiple adapters 220
deploying the adapter 218
framework overview 87
getDeviceModel method 216
getTerminalId method 217
identify device type 208
sample code 206
PvC adapter definition 330
HTTP adapter attribute 332
IP adapter attribute 333
PvC adapter attribute 333
sample 334
PvC adapter framework 16
reference information 329
PvC command
WCS command registration 263
PvC commands 16, 90, 256
create a custom command 259
deploy a custom command 259
PVCBufferURL 91
PVCBufferUrl 257, 351
PVCChangeDevice 90, 256, 347
PVCRegistration 90, 256, 340
PVCRegistrationDevice 90, 256, 344
ReEnterPassword 91, 257, 353
reference information 339
usage scenarios 257
PvC data beans 16, 91, 232
PVCBufferDataBean 91, 356
reference information 355
UserPVCDeviceDataBean 91, 356
PvC database tables
PVCBinding 364
PVCBuffer 364
PVCDEVMDL 360
PVCDEVSPEC 363
PVCMDLSPEC 361
PVCSession 364
reference information 359
PVCBinding 364
PVCBuffer 364
PVCBufferDataBean 91, 356
PVCBufferUrl 91, 102, 351
example 353
syntax 352
PVCChangeDevice 90, 347
example 350
syntax 349
PVCDEVMDL 360
PVCDEVSPEC 363
PVCMDLSPEC 361
PVCRegistration 90, 340
example 342
syntax 341
PVCRegistrationDevice 90, 344
example 346

syntax 345
PVCSession 364

Q
Qualcomm EudoraWeb 60

R
RAS 170
Redbooks Web Site 373
Contact us xv
ReEnterPassword 91, 353
example 354
Restricted command execution 98
Reverse proxy 119
REVO PLUS 58
Roaming agreement 22
Runtime environment 141
high level install steps 144
testing 142

S
Sample code defined 180
Sample store overview 181
Secure Electronic Transaction
See SET
Secure Sockets Layer
See SSL
Security
logon timeout 97
restricted command execution 98
user registration 98
Security enhancement 87
Session control 86, 92
PvC adapter 92
unique identifier 92
Session management 72
cookies 73
URL rewriting 73
Session time-out 234
Session time-out form 234
SET 130, 137
Short Message Service
See SMS
See SMS
Short Message Service Center 27
SIM 26, 134
card 27
Simulators 169
cons of using a simulator 169
pros of using a simulator 168
SmartCards 130
SMS 27, 54
defined 54
SSL 130
Symbian
EPOC 62

T
Test environments 167
using simulators 168
using simulators and gateways 168
wireless hardware
Internet 171
intranet 170
Testing 83
runtime environment 142
simulators 169
toolkits 167
Nokia Wap Toolkit V2.1 169
UP.SDK for WAP V3.2, V4.1 169
UP.SDK V3.2 for HDML 169
Toolkits and simulators 167

U
UMTS 40
Universal Mobile Telecommunications System
See UMTS
UP.SDK for WAP 273
UP.SDK for WAP V3.2 169
UP.SDK for WAP V4.1 169
UP.SDK V3.2 for HDML 169, 266
URL buffering 87, 101
URL rewriting 73
User registration form 235
UserPVCDeviceDataBean 91, 356

V
VisaNet 137
VisualAge for Java
create a package 207
create a project 207
export to a JAR 219
Voice XML 69
Voluntary site 47

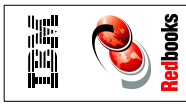
VXML 69

W

WAP 42
architecture 45
browser 60
content management scripts 274
sample code 274
sample PvC adapter code 274
sample WLM content JSPs 274
WAP stack vs Internet stack 44
WML 44
WTLS 131
WAP toolkits 272
W-CDMA 40
DS-CDMA 40
WCS V5.1
device control 86
messaging subsystem 55
programming mode 72
session control 86
session management 72
systems architecture 72
Web Clipping 52, 288
proxy 53
proxy server 61
UDP 53
WebSphere Commerce, Market Place Edition 14
WebSphere Everyplace Suite 108
components 109
m-commerce considerations 109
overview 108
WebSphere Payment Manager 136
Cassette Developer's Toolkit 137
overview 136
supported payment methods 137
WebSphere Payment Manager Framework 137
WebSphere Transcoding Publisher 115
administration 120
architecture 116
deployment model
JavaBean 120
MIME filter 120
network proxy 119
reverse proxy 119
device profiles 118
integration with WCS 300
m-commerce considerations 122
MIME filter 147
configuration 149
configuration for WTP 149
install MIME filter 146
overview 116, 296
transcoders 117
WebSphere Transcoding Toolkit 163
Profile Builder 165
Request Viewer 164
Snoop MEGlet 166
Transform Tool 163
WES
See WebSphere Everyplace Suite
Wireless Application Protocol
See WAP
Wireless LAN 40
Wireless laptop 7, 62
Wireless networks 20
cdma2000 40
cdmaOne 40
CDPD 38
cells 21
generations 23
GPRS 32
GSM 24
history 20
IMT-2000 39
Mobitex 36
PDC 39
UMTS 40
W-CDMA 40
Wireless PDA 6, 58
Wireless protocols 41
HTTP protocol 41
i-mode 46
WAP 42
Web Clipping 52
Wireless service provider 67
Wireless Transport Layer Security (WTLS) 131
WML 44, 65
WTP
See WebSphere Transcoding Publisher

X

XHTML 67



Mobile Commerce Solutions Guide using WebSphere Commerce Suite V5.1

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

Mobile Commerce Solutions Guide using WebSphere Commerce Suite V5.1

IBM m-commerce architecture and functionality

Design and development guidelines for m-commerce

Sample code for WAP WML, HDML, Palm, and WTP

This redbook provides developers and architects with the knowledge to develop and deploy m-commerce Web sites using WebSphere Commerce Suite V5.1.

This book describes the concepts related to m-commerce, such as wireless technologies, development methodology for building m-commerce Web sites, and features in WebSphere Commerce Suite V5.1 for m-commerce. Also included are integration considerations regarding IBM wireless middleware and payment options.

Detailed instructions are provided for setting up your runtime, development, and test environments. We provide a sample store and code to demonstrate IBM m-commerce functionality using WebSphere Commerce Suite V5.1.

Included are development guidelines for mobile devices directly accessing device-specific content JSPs. Samples and guidelines are included for WAP WML, HDML, Palm, and i-mode.

The book also provides design and development guidelines for mobile devices using WTP to access WebSphere Commerce Suite HTML and XML content JSPs.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6171-00 ISBN 0738422274