WebSphere. software

IBM

Patterns: SOA Client Access Integration Solutions

Process for reusing the patterns assets within RUP to design a solution

Extend the enterprise by applying Client application patterns

Patterns working example featuring IBM WebSphere Everyplace Deployment V6.0

John Ganci
Jonathan Adams
Isabella Bayer
Rebecca Chen
Daniel Ehrle
Stefan Fassmann
Jake Palmer
David Patschke
Michael Soucek

Redbooks

ibm.com/redbooks

International Technical Support Organization

Patterns: SOA Client - Access Integration Solutions

March 2006

SG24-6775-00

Note: Before using this information and the product it supports, read the information in "Notices" on page xiii.

First Edition (March 2006)

This edition applies to IBM WebSphere Everyplace Deployment V6.0, IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux, and WebSphere Everyplace Client Toolkit V6.0.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

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

- ⓔserver®
- ⓔserver®
- Redbooks (logo) ™
- eServer™
- iSeries™
- pSeries®
- xSeries®
- AIX®
- BladeCenter®
- ClearCase®
- Cloudscape™
- CICS®
- Domino®
- DB2 Universal Database™
- DB2®
- Everyplace®
- IBM®
- Lotus Notes®
- Lotus®
- MQSeries®
- Notes®
- Rational Developer Network®
- Rational Process Workbench®
- Rational Unified Process®
- Rational®
- Redbooks™
- RDN™
- RUP®
- Sametime®
- Tivoli®
- WebSphere®
- Workplace™
- Workplace Client Technology™
- Workplace Collaborative Learning™
- Workplace Managed Client™
- Workplace Messaging®
- Workplace Team Collaboration™
- Workplace Web Content Management™

The following terms are trademarks of other companies:

Enterprise JavaBeans, EJB, Java, JavaBeans, JavaScript, JavaServer, JavaServer Pages, JDBC, JDK, JSP, JVM, J2EE, J2ME, J2SE, Solaris, Sun, Sun Microsystems, Ultra, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows Mobile, Windows, Win32, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Pentium, Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

The Patterns for e-business are a group of proven, reusable assets that can be used to accelerate the design and development of e-business applications. This IBM® Redbook focuses on how to reuse the SOA Client - Access Integration patterns to extend enterprise applications to a wide range of client devices such as laptops, PDAs, and mobile phones.

Part 1, “Introduction to client access solutions” on page 1, presents the client access solution domain, and benefits of using patterns. In addition, we introduce key technologies (programming models, device types, client and server technologies, mobile connectivity and Web technologies) to serve as a foundation for understanding the Access Integration patterns.

Part 2, “SOA Client - Access Integration patterns” on page 59, guides the reader through the process of reusing the patterns assets within the context of the Rational® Unified Process® (RUP®) to accelerate the solution design. We include selection criteria to navigate from the Business and IT drivers to the appropriate Application pattern or patterns, and supporting Runtime pattern, component model, and Product mapping instantiated by IBM software products.

Part 3, “ITSO MobileAdjuster patterns working example” on page 235, demonstrates how to reuse the patterns for the fictitious ITSO Insurance scenario. The scenario highlights the Distributed Rich Client application pattern, which is used to extend the existing SOA enterprise application. The solution provides mobile adjusters with the capability to work on-site with the customer in a disconnectable mode, and sync data with the enterprise when connected to the network. The Runtime pattern is instantiated by IBM WebSphere Everyplace Deployment V6.0.

The team that wrote this redbook

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

John Ganci is a Consulting IT Specialist at the IBM ITSO, Raleigh Center, USA. John has 15 years of experience in application design, development, testing, and consulting. His areas of expertise include e-business integration, WebSphere® Application Server, WebSphere Portal, e-commerce, pervasive computing, technical team leadership, Linux®, and J2EE™ programming.

Jonathan Adams is an IBM Distinguished Engineer. He has been an IT Architect with IBM for 36 years. For the past 10 years, he has focused on helping IBM deliver reuseable end-to-end middleware solutions to its customers. Since September 1998, he has worked in the Software Group Technical Strategy organization, leading the definition and development of the Patterns for e-business. These patterns have been built by teaming across all major IBM divisions. The resultant patterns are being used by IBMers, customers, and Business Partners to help reduce risk and increase speed to market on many e-business solution developments.

Isabella Bayer is an IT Consultant and IT Architect candidate in the Pervasive Computing services group in the IBM Lab in Boeblingen, Germany. She holds a degree in Commercial Information Technology from the University of Cooperative Education in Stuttgart. Isabella has gained more than five years of experience in the mobile computing field while being engaged in various customer projects. Her areas of expertise include architectural design, development, and implementation of mobile workforce solutions, as well as in-depth energy and utility industry experience.

Rebecca Chen is a Staff Software Engineer at IBM China Development Lab in Taipei, Taiwan. She joined IBM in 2000 and has experience in software design, development, and testing. She holds a Master's degree in Computer Science and Information Engineering from National Dong Hwa University. Her areas of expertise include Global Security Toolkit, WebSphere Application Server, WebSphere Everyplace® Access, WebSphere Everyplace Deployment, pervasive computing, and networking. She has Cisco Certified Network Associate (CCNA) Certification and is also a certified Project Management Professional.

Daniel Ehrle is an IT Specialist in the pervasive computing area of the IBM Software Group in Switzerland. After he received his degree in Computer Science in 1996, he joined IBM as a Security Specialist in the security services group of IBM Global Services. He has worked as a WebSphere Technical Presales Specialist starting in 2000, and has specialized in pervasive computing since 2002. He has been engaged in many customer projects, pilots, and proof of concepts.

Stefan Fassmann is an IT Architect in the Pervasive Computing Software service group at the IBM Lab Boeblingen, Germany. After he finished his degree in Electrical Engineering in 1996, during which he specialized in radio networks, he started working on mobile workforce and remote access solutions in the mobile computing group of IBM Global Services. He changed to the Pervasive Computing Software service group in the IBM Lab in Boeblingen in 2000. Since then he has been engaged in various mobile solution customer projects using IBM WebSphere server and client software technology.

Jake Palmer is a Staff Software Engineer with IBM Application and Integration Middleware in Research Triangle Park, North Carolina. Jake joined IBM in 2003 after receiving a BSE degree in Computer Science and Electrical Engineering from Duke University. Since joining IBM, he has focused on design and implementation of applications that enable mobile workforce productivity, and the integration of these applications with enterprise data.

David Patschke is a Software Test Engineer in the IBM Software Group in Austin, Texas. He has five years of experience in the fields of embedded device and application testing. He holds degrees in Computer Science and Mathematics from the University of Texas in Austin. His areas of expertise includes client-side testing of WebSphere Everyplace Access on Windows® Mobile, Palm, and Symbian platforms. In addition, he is also versed in Device Management Administration and DB2® Everyplace server.

Michael Soucek is an IT Architect in the IBM Global Services in the Czech Republic. He has six years of experience in the field of consulting for system integration architecture for e-business. He holds a degree in mechanical engineering from Czech Technical University in Prague. His areas of expertise include System architecture, analysis and architecture of Pervasive solutions, IT Optimization consulting, technical team leadership, and system analysis of B2B and B2C solutions.

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

- Peter Kovari, Phillip Dermody, Daniel Ehrle, Stefan Fassmann, Richard Jacks, George Kroner, LindaMay Patterson, and Sami Serpola, who wrote the previous redbook edition, *Patterns: Pervasive and Rich Device Access Solutions*, SG24-6315
- Special thanks to **Jonathan Adams**, Distinguished Engineer, Project and Technical leader of the IBM Patterns for e-business, IBM UK
- Special thanks to **Jim Colson**, Distinguished Engineer, Chief Architect - IBM Client Software, IBM USA
- Keith Carter, IBM USA
- Brian Lillie, IBM USA
- Juan Rodriguez, IBM Raleigh ITSO Center, USA
- Paula Richards, IBM USA
- Ruth McCorkle, IBM USA
- Mary Fisher, IBM USA
- Salim Zeitouni, IBM USA
- Charles Levay, IBM USA

- Jim Robbins, IBM USA
- Pierre Carlson, IBM USA
- Michael Burkhart, IBM USA
- Alastair Hillis-Wragg, IBM UK
- Soheel Chughtai, IBM UK

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. 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 email to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

Part 1

Introduction to client access solutions

Part 1 presents the client access solution domain and benefits of using patterns. In addition, we introduce key technologies (programming models, device types, client and server technologies, mobile connectivity, and Web technologies) to serve as a foundation for understanding the Access Integration patterns.

1 Introduction

Too often, architects and specialists take the shortcut of simply trying to sell software products based on features rather than focusing on the customer solution. When working with a customer in a full blown services engagement or to create a proof of concept, IBM practitioners have found it to be much more effective for an IT architect to start by evaluating the business problems and build solutions to solve them.

The architect begins by gathering the requirements, developing an outline for the desired solution, and considering any special requirements that need to be factored into that solution. The architect then takes this input and designs the solution, which can include one or more computer applications that address the business problems by supplying the necessary business functions.

To improve the process over time, we need to capture and reuse the experience of the IT architects in such a way that future engagements can be made simpler and faster. We do this by capturing knowledge gained from each engagement and using it to build a repository of assets. IT architects can then build future solutions based on these proven assets. This reuse saves time, money, and effort, and helps ensure delivery of a solid, properly architected solution.

The IBM Patterns for e-business help facilitate this reuse of assets. Their purpose is to capture and publish e-business artifacts that have been used, tested, and proven to be successful. The information captured by them is assumed to fit the majority, or 80/20, situation. The IBM Patterns for e-business are further augmented with guidelines and related links.

1.1 Introduction and key concepts

The IBM software family includes many software products that can be used to extend enterprise applications and processes to a wide range of client devices, such as desktops, laptops, PDAs, mobile phones, and embedded devices. In this section we introduce the following key concepts:

- Benefits of using patterns to design a solution
- Access Integration pattern summary
- Extending SOA to the edge
- ITSO MobileAdjuster patterns working example

1.1.1 Benefits of using patterns to design a solution

The IBM Patterns for e-business help facilitate this reuse of assets to accelerate the solution design. Their purpose is to capture and publish e-business artifacts that have been used, tested, and proven to be successful.

There are several benefits to using a patterns approach to design solutions:

- Reuse of assets to accelerate the solution design

  The reuse of assets (process, Business and IT drivers, Application patterns, Runtime patterns, Product mapping, component model) can be used to accelerate the solution design. Without leveraging these reusable assets, IT architects would need to reinvent custom solutions for each customer engagement or proof of concept.

- Knowledge transfer

  Customer business processes and applications, and middleware integration requirements are increasing in complexity to meet business demands. It is virtually impossible for one IT architect to be skilled in all solution areas in which they will be designing a solution. IT architects and specialists need to be able to leverage and reuse the knowledge learned from similar engagements.

- Documented patterns used to support architectural decisions

  In many cases, the customer wants an explanation of how you arrived at the design. The Access Integration pattern represents the collective knowledge and review of many senior level architects within IBM. Having the patterns documented in a redbook provides technical validation in support of the design.

1.1.2 Access Integration pattern summary

The Access Integration pattern describes the recurring application and architecture designs that enable access to one or more Business patterns. In particular, the Access Integration pattern enables access from multiple channels (devices) and integrates the common services required to support a consistent user interface and programming model from low-function devices to rich devices.

Reusable patterns assets

This book includes the following reusable assets to accelerate the solution design:

- Process for using patterns within the Rational Unified Process (RUP)

  This book includes a process for using the Access Integration patterns within the context of the RUP. This facilities understanding how to apply the reusable patterns assets within a life-cycle methodology.

  Note: For details refer to Chapter 3, "Design a solution using the Patterns for e-business" on page 61.

- Business and IT drivers

  We have included common business and IT drivers, and capabilities to be used as criteria for selecting the appropriate Application pattern(s).

  Note: For details refer to Chapter 4, "Business and IT drivers" on page 83.

- Application patterns

  Application patterns represent the partitioning of the application logic and data together with the styles of interaction between the logic tiers. We have defined eight Application patterns specific to the client access solution domain.

  Note: For details refer to Chapter 5, "Application patterns" on page 99.

- Runtime patterns

  The Runtime pattern uses nodes to group functional requirements. The nodes are interconnected to solve a business problem. An Application pattern leads to an underpinning Runtime pattern.

  Note: For details refer to Chapter 6, "Runtime patterns and Product mapping overview" on page 129.

- Product mapping

  Each Runtime pattern has a corresponding Product mapping, which identifies the platform, software product name, and version number (IBM software products).

  Note: For details refer to Chapter 6, “Runtime patterns and Product mapping overview” on page 129.

- Component models

  Many of the Runtime patterns and Product mappings have a corresponding component model. The component models were developed using IBM Rational Software Architect. The component models can be reused in the solution design for customers using the Rational Unified Process (RUP) or other methodologies requiring component models.

  Note: For details refer to Chapter 6, “Runtime patterns and Product mapping overview” on page 129.

Summary of Access Integration application patterns

Figure 1-1 on page 7 and Figure 1-2 on page 7 provide a summary depiction of the Access Integration::Client application patterns relevant to this book. Table 1-1 on page 8 contains an overview explanation of each of the Application patterns. The function set provided by the Application patterns shown in Figure 1-1 on page 7 are incremental. The Thin Client application pattern has the lowest function set. The richest capability is delivered by the Distributed Rich Client application pattern and its variations Distributed Collaboration Client and Distributed Multimodal Client.

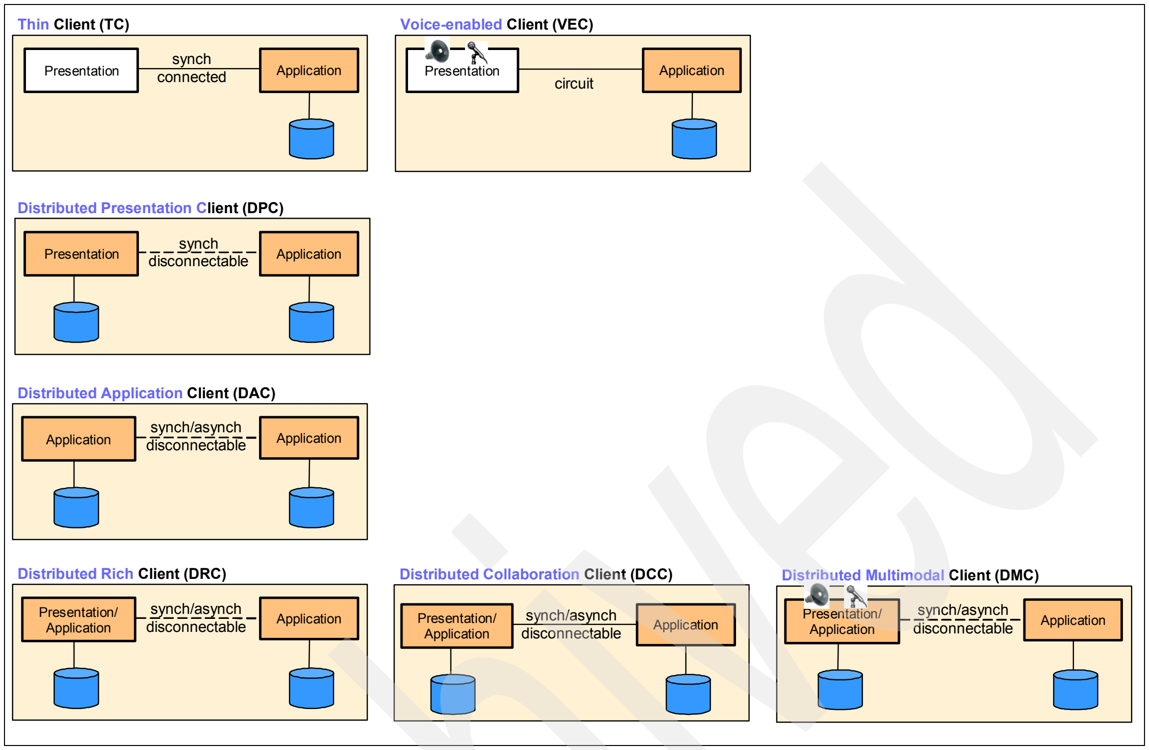

Thin Client (TC)
Presentation
synch connected
Application
Voice-enabled Client (VEC)
Presentation
circuit
Application
Distributed Presentation Client (DPC)
Presentation
synch disconnectable
Application
Distributed Application Client (DAC)
Application
synch/asynch disconnectable
Application
Distributed Rich Client (DRC)
Presentation/ Application
synch/asynch disconnectable
Application
Distributed Collaboration Client (DCC)
Presentation/ Application
synch/asynch disconnectable
Application
Distributed Multimodal Client (DMC)
Presentation/ Application
synch/asynch disconnectable
Application

Figure 1-1 Summary of Application patterns

The Built-in Client application pattern (see Figure 1-2) is a generic Application pattern that evolves into one of the other Application patterns found in Figure 1-1 depending on the function provided by the built-in application. As industry standard interfaces for basic functions of mobile devices are developed (for example, device management capabilities), it is assumed that more and more functions will be delivered by the device manufacturer as built-in applications. This will reduce the development, operations, and maintenance effort for device access solutions.

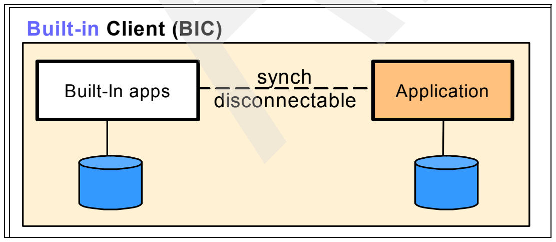


Figure 1-2 Generic Built-In Client application pattern

Table 1-1 Summary for the Access Integration::Client application patterns

Application pattern	**Description**
Thin Client (TC)	The TC application pattern includes applications that use a browser on the device to access Web-based applications synchronously in an online mode (always connected scenario). The browser is a standard interface and is usually provided with the device (for example, HTML, WML browsers).
Voice-Enabled Client (VEC)	In this case, the client is a speech user interface provided by the device (for example, phone). It interacts with a speech-enabled application over circuit switched (voice capable) networks.
Distributed Presentation Client (DPC)	The DPC represents applications that provide their own presentation layer on the device. This presentation layer is provided by the application and runs on the device. The device accesses the application in online mode, but can also work offline using a local data storage (for example, access Web content online and retrieve content from the local cache in offline mode).
Distributed Application Client (DAC)	The DAC comprises applications that run on the device without providing a user interface (remote *headless application*). Application logic is provided locally so that the application can operate independently from the back-end application logic (operation in connected and disconnected modes). The necessary data storage is also provided locally and is synchronized with the back-end data storage under-defined circumstances (time or event triggered).
Distributed Rich Client (DRC)	The DRC is a DAC with a local user interface. This demands higher system resources and implies that a user interacts directly with the application on the device. The user interface ranges from local Web-based user interfaces to more functional rich user interfaces. The DCC and DMC variations of the DRC Application pattern address specific requirements for collaboration and multimodal applications.
Distributed Collaboration Client	The DCC is a variation of the DRC Application pattern. It especially covers online collaboration requirements (for example, instant messaging), as well as disconnectable services (for example, e-mail).
Distributed Multimodal Client (DMC)	The DMC extends the DRC Application pattern with multimodal capabilities (for example, voice navigation in an application).

Application pattern	Description
Built-in Client (BIC)	The BIC is a generic form potentially supporting all of the patterns above. It addresses the fact that certain functions are provided by the device manufacturer through built-in applications. In this case, the application programmer may not need to provide middleware or applications for the device since they are provided by the manufacturer. Depending on the function of the built-in application, this pattern matches one of the other Application patterns (and corresponds to the matching Runtime pattern of the Application pattern).

Note: Each of the Application patterns has corresponding Runtime patterns, Product mappings, and component models.

1.1.3 Extending SOA to the edge

The Access Integration::Client application patterns found in this book have a common theme of extending enterprise applications to a wide range of client devices, including desktops, laptops, PDAs, mobile phones, and embedded devices. In this section, we focus our discussion on the Distributed Rich Client application pattern and its ability to extend the enterprise SOA to the edge. The Distributed Rich Client application pattern, and supporting Runtime pattern and Product mapping, is instantiated by IBM WebSphere Everyplace Deployment V6.0.

We first introduce the key concepts of SOA, and highlight the elements that make up the IBM SOA Foundation. Next, we explain how IBM WebSphere Everyplace Deployment can be used to extend the enterprise SOA to a wide range of devices.

Service-oriented architecture (SOA)

Service-oriented architecture is an architectural style for building distributed systems that deliver application functionality as services to be used for end-user applications, or for building other services. SOA brings the benefits of loose coupling and encapsulation to integration at an enterprise level.

Services are the building blocks of an SOA. Services can be invoked independently by either external or internal service consumers to process simple functions. Alternatively, the services can be chained together to form more complex functionality and to quickly devise new functionality. By adopting an SOA approach and implementing it using supporting technologies, you can build flexible systems that implement changing business processes quickly and make extensive reuse of components.

Some key attributes that define an SOA are:

- Leverages open standards to represent software assets as services
- Provides a standard way of representing and interacting with software assets
- Lets individual software assets become building blocks that can be reused in developing other applications
- Shifts focus to application assembly rather than implementation details
- Can be used internally to create new applications out of existing components
- Can be used externally to integrate with applications outside of the enterprise

Traditional integration requires technology-aware bridges between components. It is complex and expensive to implement and maintain. Program bridges are typically based on APIs and file formats, but these change and lead to instability in the integrated system.

In an SOA, each component uses the same way of talking to other components, based on platform-neutral standards. Components that understand services can talk to other services, regardless of underlying implementation. This approach greatly simplifies and strengthens integration effort.

IBM SOA Foundation

The IBM SOA Foundation is a reference architecture used to build or extend the value of existing applications and business processes. The IBM SOA Foundation includes an integrated architecture, design patterns, best practices, and solution offerings to help simplify the packaging of IBM open standards-based software for SOA.

As seen in Figure 1-3 on page 12, the SOA life cycle includes modeling the business design, assembling and deploying the information systems, and managing the deployment. Governance and processes are provided to ensure that compliance and operational policies are enforced.

The key IBM software products used in support of the IBM SOA Foundation and SOA life cycle are as follows (see Figure 1-3 on page 12):

- *Model:* IBM WebSphere Business Modeler

  IBM WebSphere Business Modeler provides easy-to-use process modeling tooling for the business analyst to help maximize process and business resource re-use.
- *Assemble:* IBM WebSphere Integration Developer

IBM WebSphere Integration Developer provides easy-to-use integration (process development) tooling to simplify and speed the assembly of composite applications. IBM WebSphere Integration Developer is built on top of Eclipse 3 technology common to the IBM Rational Software Development Platform (used by Rational Software Architect, Rational Application Developer, Rational Web Developer).

- *Deploy:* IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus (ESB)

  IBM WebSphere Process Server enables rapid assembly and deployment of business solutions by wiring reusable service components.

  IBM WebSphere ESB provides Web services connectivity, JMS messaging, and service-oriented integration for service-oriented architecture (SOA).

- *Manage:* IBM WebSphere Business Monitor

  IBM WebSphere Business Monitor provides real-time visibility into process performance, enabling process intervention and continuous improvement.

- *Extend:* IBM WebSphere Everyplace Deployment

  IBM WebSphere Everyplace Deployment provides the capability to extend SOA enterprise applications to a wide range of online and disconnectable devices such as desktops, laptops, PDAs, and mobile phones.

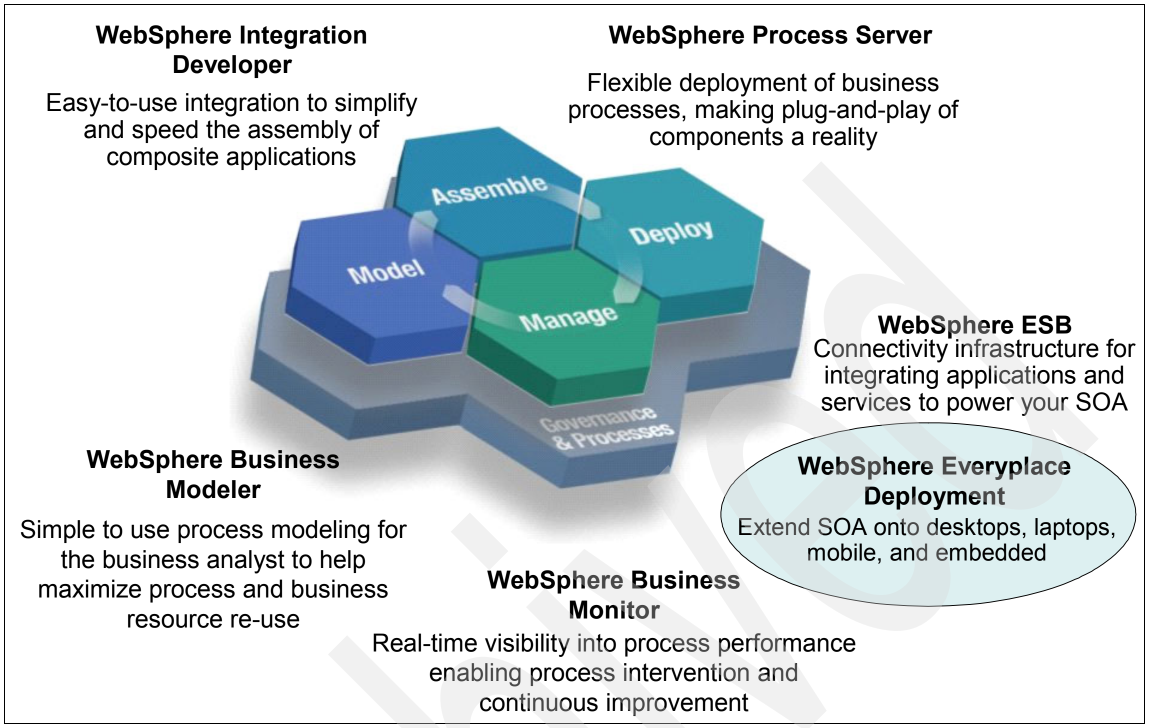

WebSphere Integration Developer
Easy-to-use integration to simplify and speed the assembly of composite applications
WebSphere Process Server
Flexible deployment of business processes, making plug-and-play of components a reality
Assemble
Deploy
Model
Manage
Governance & Processes
WebSphere ESB
Connectivity infrastructure for integrating applications and services to power your SOA
WebSphere Business Modeler
Simple to use process modeling for the business analyst to help maximize process and business resource re-use
WebSphere Everyplace Deployment
Extend SOA onto desktops, laptops, mobile, and embedded
WebSphere Business Monitor
Real-time visibility into process performance enabling process intervention and continuous improvement

Figure 1-3 IBM SOA Foundation software and SOA lifecycle

Using WebSphere Everyplace Deployment to extend SOA

This section describes the open standard components and services of IBM WebSphere Everyplace Deployment that are used to extend SOA enterprise applications to a wide range of devices, including desktops, laptops, and PDAs.

Key benefits:

- Extends SOA onto desktops, laptops, mobile and embedded devices
- Enhances traditional browser-based user interfaces
- Provides advanced user interfaces beyond the browser (Eclipse-based rich client)

Business value:

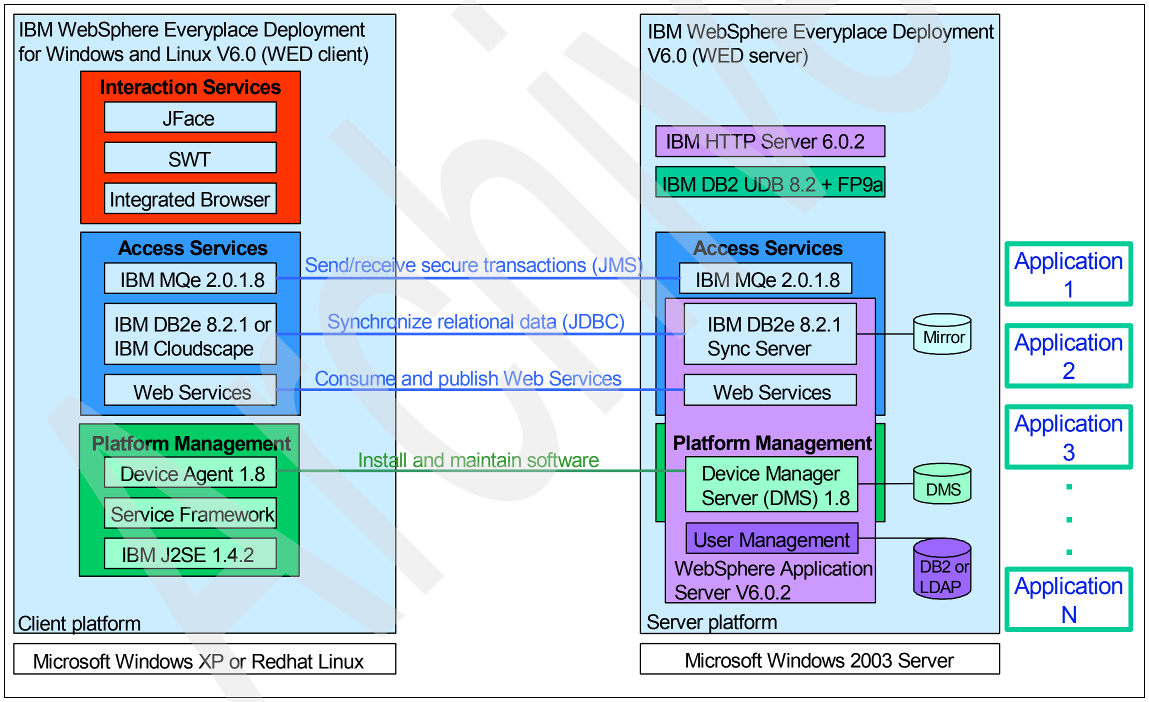
- Efficient reuse of common skills and assets all the way to the edge (devices)
- Better human and machine integration to business processes
- Increased process availability via disconnectable mode of operation

Support for SOA:

- Symmetric services out to the edge (for example, Web Services)
- Dynamically composable managed platform
- Based on open standards
- Same programming model as the rest of the SOA family

IBM WebSphere Everyplace Deployment (WED) V6.0 product is packaged into three distinct sets of software components:

- IBM WebSphere Everyplace Deployment (WED server)
- IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (WED client or ED4WL)
- IBM WebSphere Everyplace Client Toolkit (used with IBM Rational V6 tooling based on IBM Rational Software Developer - Eclipse 3)



IBM WebSphere Everyplace Deployment for Windows and Linux V6.0 (WED client)
Interaction Services
JFace
SWT
Integrated Browser
Access Services
IBM MQe 2.0.1.8
IBM DB2e 8.2.1 or IBM Cloudscape
Web Services
Platform Management
Device Agent 1.8
Service Framework
IBM J2SE 1.4.2
Client platform
Microsoft Windows XP or Redhat Linux
Send/receive secure transactions (JMS)
Synchronize relational data (JDBC)
Consume and publish Web Services
Install and maintain software
IBM WebSphere Everyplace Deployment V6.0 (WED server)
IBM HTTP Server 6.0.2
IBM DB2 UDB 8.2 + FP9a
Access Services
IBM MQe 2.0.1.8
IBM DB2e 8.2.1 Sync Server
Web Services
Platform Management
Device Manager Server (DMS) 1.8
User Management
WebSphere Application Server V6.0.2
Server platform
Microsoft Windows 2003 Server
Mirror
DMS
DB2 or LDAP
Application 1
Application 2
Application 3
Application N

Figure 1-4 IBM WebSphere Everyplace Deployment V6.0 end-to-end solution

Figure 1-4 highlights the IBM WebSphere Everyplace Deployment capabilities that provide standards-based client services for extending an enterprise SOA:

- Managed client services

The WED client platform includes the IBM Java™ 2 Standard Edition (J2SE™) 1.4.2 JVM™ (service release 2) as the base Java runtime environment for Windows XP and RedHat Linux.

The client platform provides a Service Framework that implements the OSGi R3 framework specification and provides a service-oriented architecture on top of the JVM. The OSGi framework specification is provided by the OSGi Alliance. The OSGi Alliance's mission is to specify, create, advance, and promote wide industry adoption of an open service delivery and management platform.

- Access services

  The access services include the following open standard services:

  – Use JMS to send/receive secure transactions using MQe.

  – Synchronize data from DB2e and Cloudscape™ clients to a DB2 Everyplace Sync Server.

  – Consume and publish Web Services.

- Interaction services

  The client platform is built on the Eclipse Rich Client Platform (RCP) to enable the delivery of applications that provide a rich user interface across multiple platforms. The client platform provides the Workbench, Standard Widget Toolkit (SWT), JFace, Help, and Preferences Interaction services.

- Platform Management services

  There are two key Platform Management services included in WebSphere Everyplace Deployment. First, the Eclipse Update Manager enables end-users to directly install applications and components from standard Eclipse Update Sites onto the Workbench.

  Second, the Enterprise Management Agent works cooperatively with the Device Management server provided by the WebSphere Everyplace Deployment server to perform management operations. The agent and server use the SyncML/DM protocol defined by the Open Mobile Alliance to communicate management requests. An administrator can schedule management jobs for devices that include software installation, update, and configuration.

Note: For more information about IBM WebSphere Everyplace Deployment V6.0, refer to "IBM WebSphere Everyplace Deployment" on page 439.

1.1.4 ITSO MobileAdjuster patterns working example

Part 3, “ITSO MobileAdjuster patterns working example” on page 235, demonstrates how to reuse the patterns for the fictitious ITSO Insurance scenario. The scenario highlights the Distributed Rich Client application pattern, which is used to extend an SOA enterprise application. The Distributed Application Client::Runtime pattern::Product mapping is instantiated by IBM WebSphere Everyplace Deployment V6.0 (server) and IBM WebSphere Everyplace Deployment V6.0 (client).

Note: The ITSO MobileAdjuster working example, sample application is derived from the IBM WebSphere Everyplace Deployment V6.0 MobileAdjuster sample. We have extended the application for this book and thus have packaged our own version of the sample code. For information about downloading the sample code, refer to Appendix E, “Additional material” on page 503.

The solution provides mobile adjusters with the following key capabilities to work on-site with the customer in a disconnectable mode:

- Disconnectable application with local WEB user interface
- Disconnectable application using relational database synchronization between the client and server
- Disconnectable application using transactional messaging
- Disconnectable application using Web services (consume and publish)

The ITSO Insurance scenario walks the reader through a end-to-end working example including the following chapters (topics):

- Chapter 8, “Requirements analysis and solution design” on page 197
- Chapter 9, “Application design” on page 237
- Chapter 10, “Development environment installation” on page 255
- Chapter 11, “Application development” on page 279
- Chapter 12, “Runtime environment installation” on page 325
- Chapter 13, “Application deployment” on page 345
- Chapter 14, “Managing clients” on page 375
- Chapter 15, “Application walkthrough” on page 407

1.2 Target audience of book

This book includes architecture, design, development, integration, deployment, and administration topics. The target audience of the book can best be matched by role to the topic of interest within the publication.

1.2.1 Roles and skills

This section includes a brief description of the following user roles defined for the target audience for this book.

- Business Analyst
- Solution Architect
- Web Developer
- Enterprise Bean Developer
- RCP User Interface Developer
- Application Integrator
- Solution Deployer
- End User
- Platform Provider
- Database Administrator
- IT Administrator

Note: The user roles described in this section are based largely on a subset of roles found in the IBM Software Group User Roles V1.0 specification.

Business Analyst

The Business Analyst role identifies customer and business needs, captures requirements (often in the form of natural language, or free-form text), creates business use cases, and looks for areas of optimization—especially those that can be automated by software. As the name implies, the Business Analyst is focused on the customer needs as well as on the development organization's ability to fill the need with a software solution. The Business Analyst often is a line-of-business departmental expert who will define (and test) business rules and policies. The goal is to provide a comprehensive and structured view of the business domain.

Table 1-2 Business Analyst role responsibilities and skills

Responsibilities	Skills
► Identify business goals. ► Analyze the business for a product or line of products. In some cases determine product goals, competitive threats, and pricing. ► Assess target organization; understand the end users and their domain. ► Capture business vocabulary of the target user. ► Set and adjust project objectives. ► Define business architecture/domain model. ► Structure the business use case model. ► Find and detail business use cases and actors. ► Identify business processes that can be automated or optimized, and define automation requirements. ► Ensure final application meets requirements.	► Expert in identifying and understanding problems and opportunities ► Expert in working with end users to understand their goals, constraints, and needs, while identifying areas for improvement through software ► Ability to articulate the needs that are associated with the key problem to be solved or opportunity to be realized ► Knowledge of the business domain ► Comfortable with business architecture, business modeling, and business

Solution Architect

The Solution Architect is responsible for the design and coordination of a new solution, application, or component, based on the specified functional (IT infrastructure) and non-functional business requirements, and has overall responsibility for driving the major technical decisions.

They usually create and maintain whole systems architectures (including hardware and software) like missile defenses systems or automobiles (industrial), and create and maintain the overall structure and layout of the enterprise IT systems, including:

- The interface of new business services.
- IT strategies and expected behavior of service in terms of performance, service levels, etc.
- The task flows for enabling the business service.

Within this book, the Solution Architect is responsible for analyzing the customer requirements, selecting the appropriate Application pattern or patterns, and supporting Runtime pattern and Product mapping to create the solution design.

Table 1-3 Solution Architect role responsibilities and skills

Responsibilities	Skills
► Model software applications and solutions. ► Write application design for a software system's components and their interfaces within and outside of the software system, including clear specifications of functional and non-functional requirements for the solution. ► Modify existing services to react to new requirements. ► Design the optimum IT infrastructure for a given business solution. ► Define expected behaviors of service in terms of performance, service levels, etc. ► Prioritize use cases. ► Construct and assess viability of architectural proof-of-concept. ► Participate in IT architecture planning. ► Acts as Solution Champion: – Maintains internal business and vendor relationships – Supports Sales and Marketing teams through demonstrating solutions to potential customers and educating on the solution's particular features	► Understands business needs/goals of company or group ► Understands overall technical goals of the business and the current hardware/software inventory ► Understands industry patterns and approaches related to solution ► Understands solutions or technologies that map to company needs as well as prerequisites and hardware/software requirements ► Good overview of company's entire business and IT operations. ► Very knowledgeable about one or more particular solutions ► Good communication skills ► Able to champion a project across organizational boundaries and with potential end users

Web Developer

The Web Developer develops Web components that run on the WED client platform. The Web Developer leverages their existing J2EE Web development skills to build their applications. They also can reuse existing Web components.

Table 1-4 Web Developer role responsibilities and skills

Responsibilities	Skills
► Creates Web components ► Converts J2EE Web components ► Migrates Workplace™ Client Technology Micro Edition 5.7.1 Web components ► Selects Platform Profile for target ► Specifies deployment information ► Writes servlet source code ► Writes JSP™ and HTML files ► Optionally uses Struts or JSF ► Compiles Web components ► Packages the components in a Web Application Bundle (WAB) file ► Unit tests and debugs components ► Imports/exports WABs	► Experienced with developing servlets, JSPs, HTML, and Java components ► Optionally skilled in Struts or JSF ► Skilled in using standard services including database access (JDBC™), messaging (JMS), and Web Services ► Understands and uses the ED4WL tools and possibly RWD/RAD to build Web components for target devices

Enterprise Bean Developer

The Enterprise Bean Developer develops Embedded Transaction Applications (ETAs) that run on the WED client platform. The Enterprise Bean Developer leverages their existing J2EE EJB™ development skills to build their applications. They can also reuse existing EJB components that comply with the Embedded Transaction Container (ETC).

Table 1-5 Enterprise Bean Developer role responsibilities and skills

Responsibilities	Skills
► Creates Embedded Transaction Applications ► Converts J2EE EJBs ► Selects Platform Profile for target ► Specifies deployment information ► Writes and compiles the source code ► Packages components in Embedded Transaction Application files ► Unit tests and debugs components ► Imports/exports Embedded Transaction Applications	► Experienced with developing EJBs ► Skilled in using standard services including database access (JDBC), messaging (JMS), and Web Services ► Understands and uses the ED4WL tools and possibly RAD to build Embedded Transaction Applications for target devices ► Understands the differences between the complete EJB spec and ETC

RCP User Interface Developer

The RCP User Interface Developer is responsible for developing GUI components that run on the WED client platform. The RCP User Interface

Developer leverages their existing Eclipse RCP UI development skills to build these components. They also can reuse existing RCP UI components.

Table 1-6 RCP User Interface Developer role responsibilities and skills

Responsibilities	Skills
► Creates RCP user interface components ► Selects Platform Profile for target ► Writes and compiles the source code ► Uses the required extension points to register applications with ED4WL and integrate with the ED4WL workbench (desktop), helps, and preferences ► Specifies the plugin.xml file ► Packages the .class files and plugin.xml file in a plug-in JAR file ► Unit tests and debugs applications	► Understands the end users and their tasks ► Skilled in developing user interfaces to achieve requirements ► Experienced with developing user interface components with the RCP ► Understands and uses the ED4WL tools and possibly the Eclipse Visual Editor to build RCP user interface components for target devices

Application Integrator

The Application Integrator assembles components into applications.

Table 1-7 Application Integrator role responsibilities and skills

Responsibilities	Skills
► Assembles application components into Eclipse Features. Resolves inter-dependencies. Specify prerequisites. ► Creates licenses for assembled Features. ► Provides instructions that describe external dependencies of the features that the Solution Deployer must resolve in the deployment process. ► Executes a simple path through the application/solution to validate it. ► Verifies that the feature can run on the Platform Profiles supported by the target devices.	► Good understanding of the business solution and the business environment ► Knowledge of application/solution components and their artifacts in order to enable their efficient collaboration at run time ► Understands and uses the application integration tools

Solution Deployer

The Solution Deployer configures and deploys applications.

Table 1-8 Solution Developer role responsibilities and skills

Responsibilities	Skills
► Checks for and installs dependencies. ► Configures applications for operational environment (for example, a port). ► Deploys applications and components via Update Manager. Creates Update Sites and deploys features onto Update Sites so end users can install features onto ED4WL. ► Deploys applications and components via DMS - Package an Update Site into an OSGi bundle with a flag set in the bundle manifest that indicates this bundle contains an Update Site so the Enterprise Management Agent installs the features in the Update Site onto ED4WL. Places bundles in DMS repositories. ► Executes a simple install from the Update Site or DMS to verify.	► Understands platforms being deployed to ► Ability to determine compliance of deployed solution ► Ability to troubleshoot deployment problems ► Ability to build and test releases and write deployment scripts ► Understands and uses deployment tools ► PDE - Update Site Project ► NativeAppBundle tool ► EJB deployment tools

End User

The End User installs, manages, and launches applications on WED client.

Table 1-9 End User role responsibilities and skills

Responsibilities	Skills
► Installs the ED4WL client platform onto their desktop or laptop ► Installs optional ED4WL components ► Launches ED4WL client platform ► Installs one or more applications ► Launches applications from the ED4WL desktop ► Updates applications ► Migrates applications from WCTME-EO 5.8.1 to ED4WL ► Uninstalls applications ► Uninstalls the ED4WL platform ► Manages bundles/plug-ins ► Changes/configures runtime environment settings ► Submits problem log for support	► Understands how to download software onto operating system or use a CD ► Knows how to launch InstallShield ► Knows how to launch ED4WL from the operating system ► Understands Update Manager ► Understands how to configure the Enterprise Management Agent ► Understands how to navigate the ED4WL desktop ► Knows how to launch applications from the ED4WL desktop ► For advanced end user, knows how to manage installed applications ► Uses preferences to manage settings ► Knows how to gather eclipse log data

Platform Provider

The Platform Provider delivers a WED client platform for client devices.

Table 1-10 Platform Provider role responsibilities and skills

Responsibilities	Skills
► Defines one or more Platform Profiles, which specify the components and applications for each platform ► Builds platform-specific file that can be installed into device with components and applications ► Provides an Update Site for optional components and/or updates.	► Understands operating system and processor combinations, connectivity capabilities, and file formats for target nodes ► Knows how to package components to run on ED4WL ► Knows how to create and distribute Platform Profiles ► Understands and uses the ED4WL Platform Builder tool ► Understands the PDE - Update Site

Database Administrator

The spectrum of Database Administrators (DBA) is manifold, varying according to the use of the databases to be administered, for example:

- Traditional Database Administrator: Responsible for database performance, data integrity, database security, and database troubleshooting
- Replication Administrator: Responsible for setting up and managing replication processes (including transformation of data if needed), replication performance, data integrity, replication security, and replication troubleshooting
- Data Warehouse Administrator: Responsible for setting up and managing data movement processes (including transformation of data if needed), datawarehouse performance, data integrity, datawarehouse security, and datawarehouse troubleshooting

Table 1-11 Database Administrator role responsibilities and skills

Responsibilities	Skills
► Create and manage database synchronization and subscriptions. ► Monitor performance and tune parameters to provide fast query responses by end users ► Monitor database server resources: Memory, CPU utilization, I/O, etc. ► Manage storage and maintenance of DB objects such as tables, views, triggers, stored procedures, data types, functions, XML schemas, etc. ► Write scripts and stored procedures to manage DB. ► Perform database maintenance such as backup or reorganization. ► Set up and manage database access and privileges (for example, drop privileges). ► Create, configure, tune, monitor, and perform troubleshooting for one or more databases, replication processes, or datawarehouses/datamarts. Generally manage databases of one type (for example, DB2 only). ► Verify integrity of replicated data. ► Update data warehouse and data replication processes as changes are made to underlying objects (such as tables).	► General operations administrator skills ► Specific administration skills for a specific type of database, but may not have the skills for changing the database structure ► Specific administration skills for specific types of replication or data warehouse products ► For troubleshooting, understanding which piece, including competitor's products. may be causing the problem, especially for replication.

IT Administrator

The IT Administrator is responsible for performing all operational processes and procedures, ensuring that all IT services and infrastructure meet operational targets. Specialized types of IT Administrators include:

- System Administrator: Responsible for applying required maintenance to systems and software, keeping them up and running and current.
- Network Administrator: Concerned with configuring, managing, and maintaining operational availability of the network, including switches, routers, domain servers, connectivity, and performance. May also install and maintain network resources.

- Web Administrator: Responsible for ensuring proper configuration, capacity, and maintenance of the Web environment.
- Storage Administrator: Installs and maintains storage software and/or hardware; ensures data integrity and data availability; serves as level 2 support.
- Event Administrator: Installs, configures, and maintains event management software, ensuring the operations staff has effective tools to do their jobs; defines correlation rules for events.
- Development Tools Administrator: Responsible for the supporting tools on the development project, including selecting and acquiring tools, configuring and setting up the tools, and verifying that the tools work; also responsible for providing the overall software code Configuration Management (CM) infrastructure and environment to the product development team.
- Security Administrator: Manages and maintains the security system, bringing all security applications together.
- Automation Administrator: Concerned with product-specific and cross-product solution-level automation.

Table 1-12 IT Administrator role responsibilities and skills

Responsibilities	**Skills**
► Manage and maintain the IT environment and its operations management tools, including system administration, security standards, networking, backup, and maintenance of servlet frameworks for re-use within other solutions. ► Automate all of the following: Display of resource status, communications facilities, message scheduling and database operations, level 1 diagnosis of system errors, type of processing following errors, starting and stopping of resources, start up of VTAM, and TCP/IP network. ► Further, targeted or specialty administration is carried out by administrative specialists in those areas (for example, WebSphere, security, network, platforms, events, etc.)	► A good understanding of the specific hardware and software components and the possible dependencies between these components ► Specialty training on servers, platforms, systems, etc. ► General operator skills ► Understanding of the underlying processes used by the project ► Automation handling skills ► Good understanding of capacity/performance issues within and across systems

1.2.2 Matching book topics to roles and skills

Table 1-13 provides a summary of the book topics by part and chapter/appendix for the defined roles and skills.

Table 1-13 Matching book topics to roles and skills

	Chapter/appendix	Primary	Secondary
Part 1, "Introduction to client access solutions"			
	Chapter 1, "Introduction" on page 3	All user roles	
	Chapter 2, "Technology options" on page 27	Solution Architect	All user roles
Part 2, "SOA Client - Access Integration patterns"			
	Chapter 3, "Design a solution using the Patterns for e-business" on page 61	Solution Architect	All user roles
	Chapter 4, "Business and IT drivers" on page 83	Solution Architect	All user roles
	Chapter 5, "Application patterns" on page 99	Solution Architect	All user roles
	Chapter 6, "Runtime patterns and Product mapping overview" on page 129	Solution Architect	All user roles
	Chapter 7, "Runtime patterns and Product mapping details" on page 155	Solution Architect	All user roles
Part 3, "ITSO MobileAdjuster patterns working example"			
	Chapter 8, "Requirements analysis and solution design" on page 197	Business Analyst Solution Architect	All user roles
	Chapter 9, "Application design" on page 237	Solution Architect Web Developer EJB Developer RCP UI Developer	Application Integrator
	Chapter 10, "Development environment installation" on page 255	Web Developer EJB Developer RCP UI Developer Application Integrator	Solution Architect DBA IT Administrator
	Chapter 11, "Application development" on page 279	Web Developer EJB Developer RCP UI Developer Application Integrator	Solution Architect DBA
	Chapter 12, "Runtime environment installation" on page 325	Application Integrator DBA IT Administrator	Solution Architect

	Chapter/appendix	Primary	Secondary
	Chapter 13, “Application deployment” on page 345	Solution Deployer IT Administrator DBA	Solution Architect Web Developer EJB Developer RCP UI Developer Application Integrator
	Chapter 14, “Managing clients” on page 375	Solution Deployer IT Administrator DBA	Solution Architect Application Integrator
	Chapter 15, “Application walkthrough” on page 407	End User Business Analyst Solution Deployer	All user roles
Part 4, “Appendixes” on page 431			
	Appendix A, “IBM software product descriptions” on page 433	Solution Architect	All user roles
	Appendix B, “Description of IT drivers and capabilities” on page 465	Solution Architect	All user roles
	Appendix C, “Rational Unified Process (RUP) overview” on page 483	Solution Architect	All user roles
	Appendix E, “Additional material” on page 503	All user roles	

2

Technology options

This chapter provides readers less familiar with the access-related technology options with a technical foundation for understanding the Client Access Integration patterns. Many of the chapters to follow in this book make references to programming models, client device classes, client platforms, server and client technologies, connectivity, and mobile Web technologies discussed in this chapter.

This chapter is organized into the following sections:

- Programming models
- Client devices classes and platforms
- OSGi
- Server technologies
- Connectivity technologies
- Mobile Web technologies

2.1 Programming models

This section highlights the key components and services of three programming models relevant to the Access Integration patterns found in this book. First, we outline the elements of the conventional J2EE Web programming model as a base for discussion. Second, we introduce the Rich client Web UI programming model. Third, we describe the Rich client programming model.

The conventional J2EE Web programming model requires an online Web browser client. We describe how the Rich client Web UI and Rich client application programming models can be used to extend the conventional J2EE Web programming model to the edge with the ability to run in a disconnectable mode as well as online. The edge refers to a wide range of devices such as high-function desktop or laptop systems, and lower function devices such as PDAs, mobile phones, and embedded devices.

2.1.1 Conventional J2EE Web programming model

In the conventional e-business end-to-end programming model, also known as the J2EE Web programming model, an end-to-end application consists of components distributed across multiple tiers and nodes, where each node is a physical platform such as a device or server. These components can be logically classified by using the popular model-view-controller (MVC) pattern. In MVC, a model represents business data and the business logic that manipulates that data, a view interacts with the model on behalf of a user or other external interaction, and a controller coordinates the flow of interaction between the view and the model.

As you can see in Figure 2-1 on page 29, the architecture is based on the Model-View-Controller (MVC) pattern running on the server. The MVC pattern ensures separation of business logic from presentation logic and consists of the following three parts:

- *Model:* The model represents the data and business logic of the application. The model layer manages the application domain’s concepts, both behavior and state. It responds to requests for information about its state and responds to instructions to change its state.
- *View:* The view layer implements a rendering of the model. The responsibility of the view is to know what parts of the model’s state are relevant for the user, and to query the model for that information. The view retrieves the data from the model or receives it from the controller, and renders the content to be displayed to the user in a way the user expects to see it (for example, Web browser).

- *Controller:* The controller's responsibility is to capture user events and to determine which actions each of these events imply, depending on both the user's and the application's state. This usually involves verifying pre and post conditions. These actions can then be translated to messages to the model and view layers, as appropriate.

The components of the application run in containers located on tiers as defined by the J2EE architecture. Web applications run in a Web Container on the Presentation Tier and Enterprise JavaBeans™ (EJBs) run in an EJB container on the Business Tier. Containers can provide Interaction services that enable J2EE components to interact with users and Access services that enable J2EE components to access Enterprise applications, services, and data.

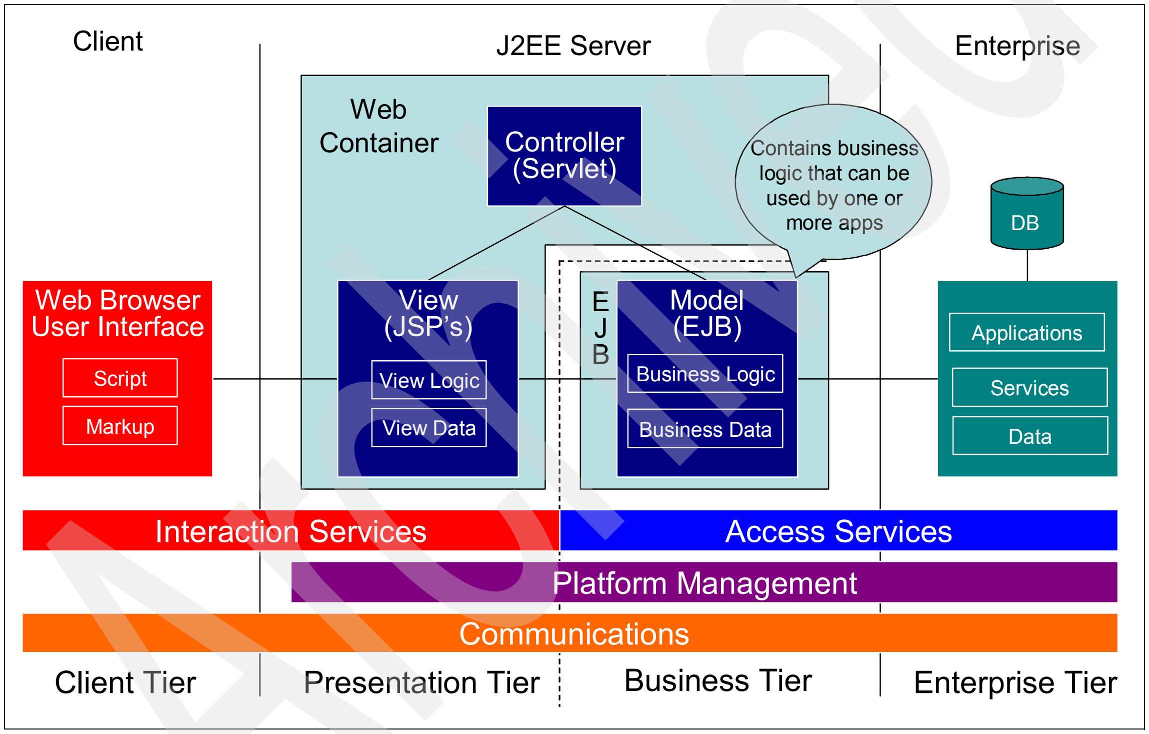


Figure 2-1 Conventional J2EE Web programming model

Figure 2-1 highlights the services common to the conventional J2EE Web programming model:

- *Interaction services:* Interaction services provide the user interface to the information. In the J2EE Web programming model, a Web browser is the user interface to display the results of the view.

- *Access services:* Access services provide access to enterprise applications, services, and data (for example, EJBs talking to back-end applications via Web services).
- *Platform Management:* Platform Management provides the ability to centrally manage (deploy, configure, start, stop, update, and delete) applications and components.
- *Communication:* The communication components provide connectivity between client, presentation, business, and enterprise tier. They ensure that remote clients have secure and seamless access to their server parts.

2.1.2 Access - Extend the J2EE Web programming model

The conventional J2EE Web programming model is designed for users who access online applications via a Web browser from high function devices such as a desktop or laptop. The device sends requests to the Web application, and the Web application receives the responses over a relatively reliable, high-bandwidth network. The J2EE Web programming model does not, however, accommodate users who work in a much more diverse environment in which these fundamental assertions do not hold true.

The IBM Server Managed Client technology is designed to extend the J2EE Web programming model to a wide range of devices, connection states, and user experiences, while allowing J2EE developers to use their existing skills.

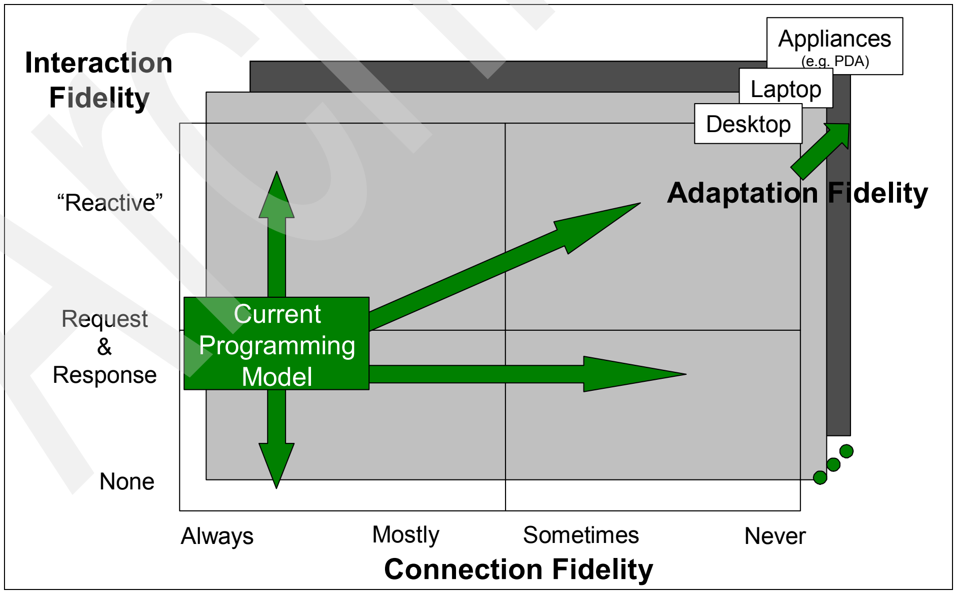


Figure 2-2 Access - Extend the J2EE Web programming model

Extending the J2EE Web programming model leads to unique requirements in the following three areas (see Figure 2-2 on page 30):

- *Connection Fidelity:* The programming model must enable applications to operate across states that are always, mostly, occasionally, and never connected. We use the term *disconnectable* to represent the set of states. Network characteristics determine the active connection state. These characteristics range from fully connected, high-bandwidth connections such as wired Ethernet, to very low bandwidth connections such as today's 2G cellular networks. Network characteristics also include latency, geographic coverage, and even billing models.
- *Interaction Fidelity:* The programming model must continue to support the dominant Web browser-based request and response user experience. However, the programming model must also support other user experiences that are increasingly becoming important. Rich user experiences include rich graphical user interfaces (GUIs) that directly utilize the native widgets of an operating system and multimodal browsers that combine visual and voice interactions in a single user experience. For embedded devices, no user experience is available; however, the programming model must enable applications to run on those devices.
- *Adaptation Fidelity:* The programming model must address a wide range of client devices covering desktops, laptops, tablets, and appliances, which include embedded controllers, smart phones, and PDAs.

Note: The IBM Server Managed Client technology is instantiated by IBM WebSphere Everyplace Deployment V6.0 (server) and IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (client).

The IBM Server Managed Client technology defines a programming model that addresses these requirements through a set of containers and services that enable applications to operate in this environment. This includes the capability to distribute key components of applications to the client, and possibly intermediate tiers, so users can continue to perform selected business operations with these applications even when their device is offline.

Moving application components to the client can also result in improved performance through local interactions between the end user and their application, and alternate view/control choices, such as a graphical user interface. By preserving a selected set of J2EE APIs and roles, developers can reuse their skills and experience to develop client applications.

2.1.3 Rich client Web UI programming model

The Rich client Web UI programming model (see Figure 2-3) extends the J2EE Web programming model by providing an Embedded Web Container and Embedded Transaction Container (ETC) on the device. The containers and services support selected APIs from the J2EE programming model to facilitate reuse of components and skills, while also supporting online and offline operations (disconnectable applications).

The Embedded Web Container extends the J2EE and WebSphere programming model to the client by supporting standard Web applications that comply with the Servlet 2.3 and JSP 1.2 specifications as well as the Servlet 2.4 and JSP 2.0 specifications. The Embedded Transaction Container further extends the symmetry of the client programming model with the J2EE and WebSphere programming model by supporting a subset of the EJB 2.0 specification.

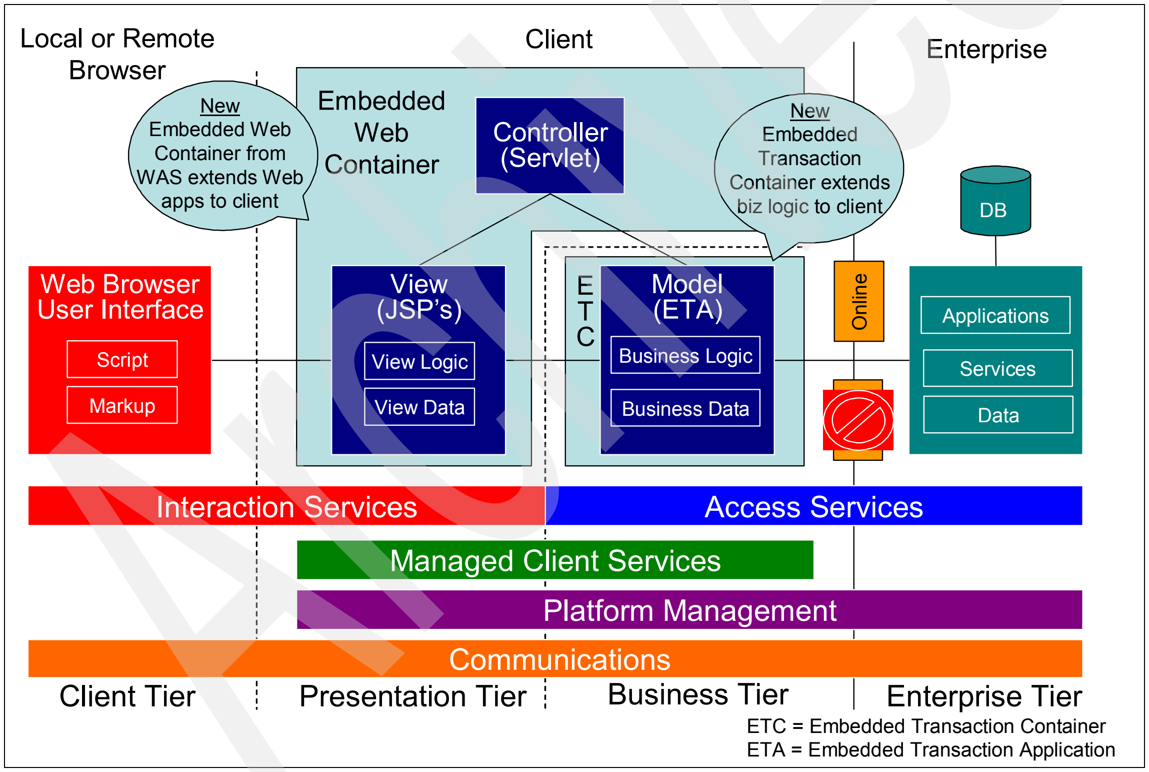


Figure 2-3 Rich client Web UI programming model

Note: The Rich client Web UI programming model is instantiated by IBM WebSphere Everyplace Deployment V6.0 (server) and IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (client).

The main client services that extend the conventional J2EE Web programming model to the Presentation and Business tier on the device are as follows.

- *Interaction services:* Interaction services in the Rich client Web UI programming model provide the same kind of user interaction as the conventional J2EE Web programming model, which allows access the view via a Web browser. This is provided by using an Embedded Web Container on the client, and the use of a remote or local Web browser.
- *Access services:* As is the case for the conventional J2EE Web programming model, the access services for the Rich client Web UI programming model are responsible for providing access to back-end applications and services.

  The combination of the Embedded Web Container and Embedded Transaction Container (ETC) allow enterprise transaction applications (ETAs) to be deployed and run on the mobile device. This concept also provides access services such Web services (publish and subscribe), database synchronization, and messaging components.
- *Managed client services:* The Managed client services provide a secure and self-contained runtime environment on the client device. Applications originally developed using the J2EE Web programming model can be easily adapted to run on a single JVM instance on the client using the *fit for purpose* class libraries (use to address device resource constraints).

  This runtime environment is essential for securely running transactional components, applications, and services outside the traditional server-side boundaries. Managed client services deliver the ability to deploy, modify, remove, and run applications and services on the client platform independent of the client operating system.
- *Platform Management:* Platform Management contains a server and client component, which is included in the Managed client services runtime environment. The Platform management client, known as the agent, interacts with the Platform management server to manage the life cycle of applications and services. Some common management tasks include device configuration, inventory collection, and software distribution.
- *Communications:* Communications services ensure secure, seamless connectivity between the client and the back-end services. The Communication services are transparent to the other services and might

contain features for working in mixed online and disconnected environments. We have listed the key Communication services:

- Seamless network roaming
- Traffic optimization (for example, compression)
- Session management (for example, maintaining session in case of short physical connection losses)
- Security (for example, encryption and two-way authentication)

2.1.4 Rich client programming model

The Rich client programming model (see Figure 2-4 on page 35) extends the J2EE Web programming model in much the same way as the Rich client Web UI programming model, with the addition of a rich graphical user interface provided by the *Eclipse*[1] Rich Client Platform (RCP). Eclipse RCP includes operating system specific graphic libraries and extensions. You can continue to use business logic components, such as Embedded Transaction Applications (ETAs), with your rich client applications.

This platform also includes Access services, Managed client services, and Platform Management. This extends the management of applications to a common graphical user interface. Client-side applications following the MVC pattern are controlled and managed within the common Eclipse (runtime) platform.

1 Eclipse - open source community whose projects are focused on providing an extensible development platform and application frameworks for building software (http://www.eclipse.org)

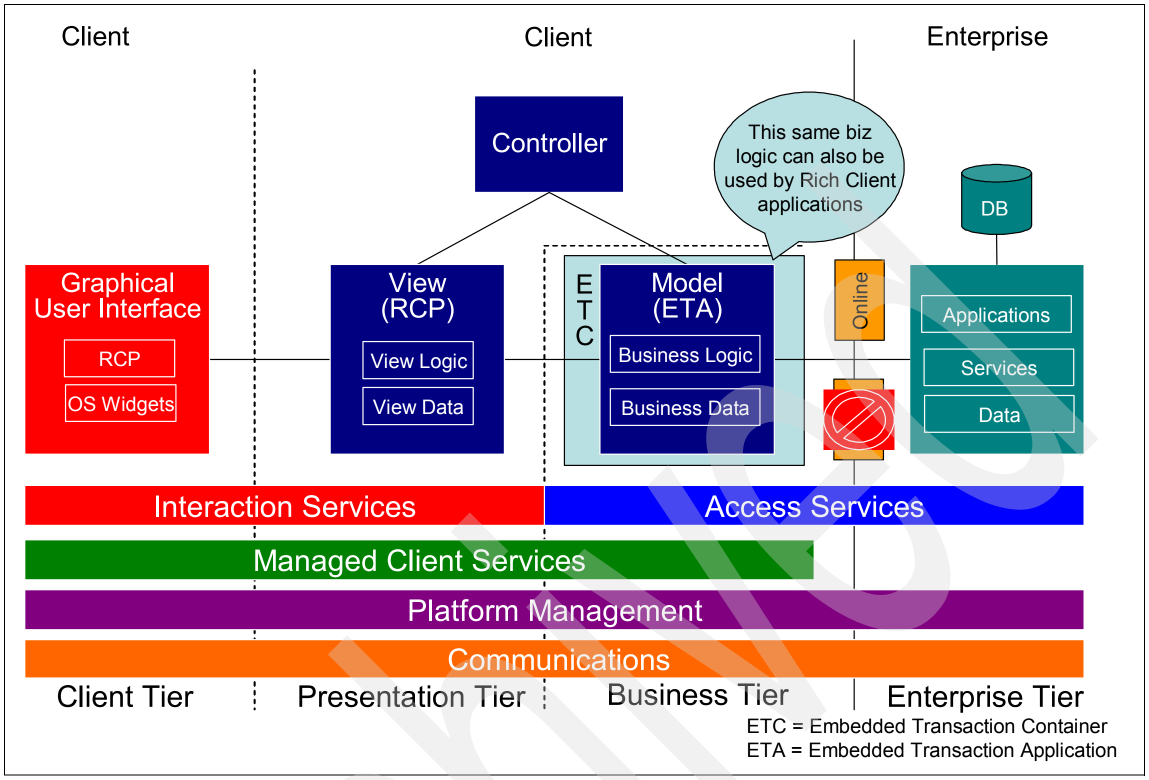


Figure 2-4 Rich client programming model (Eclipse-based rich client UI)

Like the Rich client Web UI programming model, the Rich client programming model extends the J2EE Web programming model with the following services on the device:

- *Interaction services:* In addition to the Interaction services found in the Rich client Web UI programming model, the Rich client programming model is extended with graphical support for rich user experience using Eclipse (SWT, AWT, Eclipse Workbench, and RCP).
- *Access services:* The Access services are the same as the Rich client Web UI programming model.
- *Managed client services:* The Managed client services are the same as the Rich client Web UI programming model.
- *Platform Management:* Platform Management is the same as the Rich client Web UI programming model.
- *Communications:* Communications are the same as the Rich client Web UI programming model.

2.2 Client devices classes and platforms

In this book we distinguish between client device classes and client platforms. This section provides a description and examples for both client device classes and platforms as a basis for understanding elements of the patterns.

2.2.1 Client device classes

A client device class represents a common set of features and functions provided by the device manufacturers and providers. The client device class has very specific physical hardware characteristics, capabilities, and limitations. We do consider the implications of the client device class as selection criteria for identifying the appropriate patterns later in this book.

A growing variety of client devices differ in capabilities such as input, output, processing, communication options, formfactor, etc. In some cases, the customer defines the device to be used for the solution. In other cases, you have the ability to select the type of device based on the requirements and capabilities needed.

Although the capabilities of the devices are not always easily defined, we have categorized the client devices into the following classes for reference purposes (see Figure 2-5 on page 37):

- Full function (FF)
- High function (HF)
- Limited function (LF)
- Specialized function (SF)

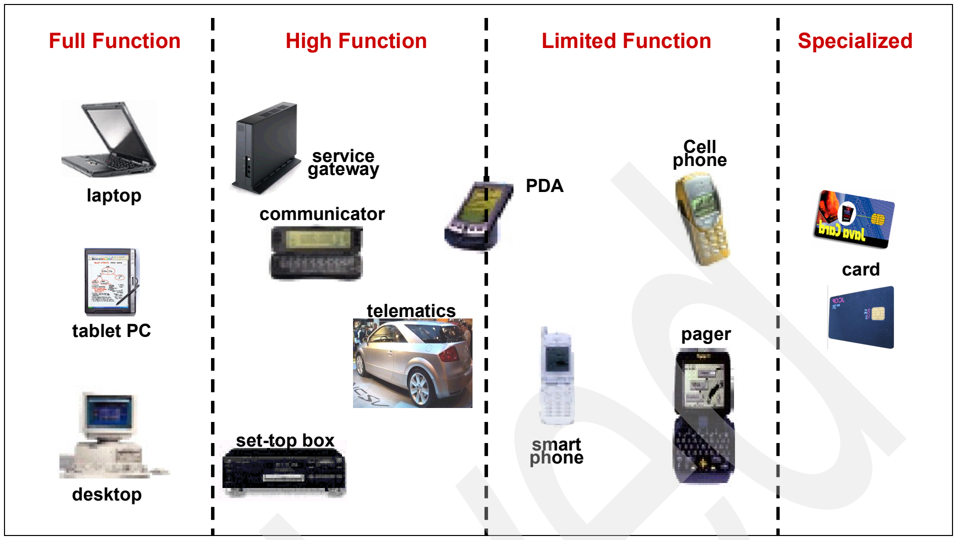


Figure 2-5 High-level client device classes

Full function (FF)

Full function client devices provide the most capable device class by supporting the most criteria outlined for the device selection process. For example, they offer user-friendly input and output options, high system capability (fast processor, at least 256 MB RAM, hard disk storage, etc.), and communication channels.

Full function devices are capable of running a rich client framework and support selected programming models.

Some common examples of full function devices include desktop, laptop, and tablet PCs. These devices typically include a high resolution screen, keyboard or touch pad, large capacity hard disk (rather than flash memory), and broadband communication options such as wired Ethernet, WiFi, etc.

High function (HF)

High function client devices include many essential criteria used during device selection, but with some limitations. For example, high function devices have a more limited user interface, less communication options, and limited system capability. High function devices may be able to store significant amounts of data in non-volatile memory. However, PDAs, for example, may only have volatile memory on-board, or use flash-based memory.

High function devices are capable of running a rich client framework and support selected programming models.

Some common examples of high function devices include service gateways, set-top boxes, high-end PDA with a comprehensive input/output options and communication capabilities (for example, WiFi, Edge/GPRS, GSM), and telematic devices.

Limited function (LF)

Limited function client devices represent a vast number of typical mobile end-user devices such as mobile phones, smart phones, and basic PDAs. This type of device has a limited user interface, limited or no connection options, and low to medium system performance.

A common example of a limited function device is a mobile phone with a browser and a proprietary operating system. The type of device has a limited screen size and basic input capabilities.

Specialized function (SF)

Specialized function client devices are embedded devices with no real periphery. They provide only very basic input/output options, no direct user interface, and very basic communication capabilities (for example, magnetic indication impulse).

Some common examples of the specialized function device class include smart cards, RFID tags, and sensors.

2.2.2 Client platforms

The client platform defines the frameworks and common services that collectively make up an integration layer to support the common programming models on the client devices. The platform is defined as a combination of the physical device and the framework running on the device.

We have defined the following client platforms to align with the Application patterns found in Chapter 5, “Application patterns” on page 99:

- Thin Client (TC)
- Voice-Enabled Client (VEC)
- Distributed Presentation Client (DPC)
- Distributed Application Client (DAC)
- Distributed Rich Client (DRC)
- Distributed Collaboration Client (DCC)
- Distributed Multimodal Client (DMC)
- Built-in Client (BIC)

Thin Client (TC)

A Thin Client follows the client-server paradigm and is designed to serve a client with a Web browser user interface for online access to Web applications. The Thin Client requires limited system resources in which the server component does the work and the content is rendered for display in the Web browser. The Thin Client is common to desktop or laptop devices with an HTML Web browser (Microsoft® Internet Explorer or Mozilla Firefox), or a PDA or mobile phone with a WAP browser.

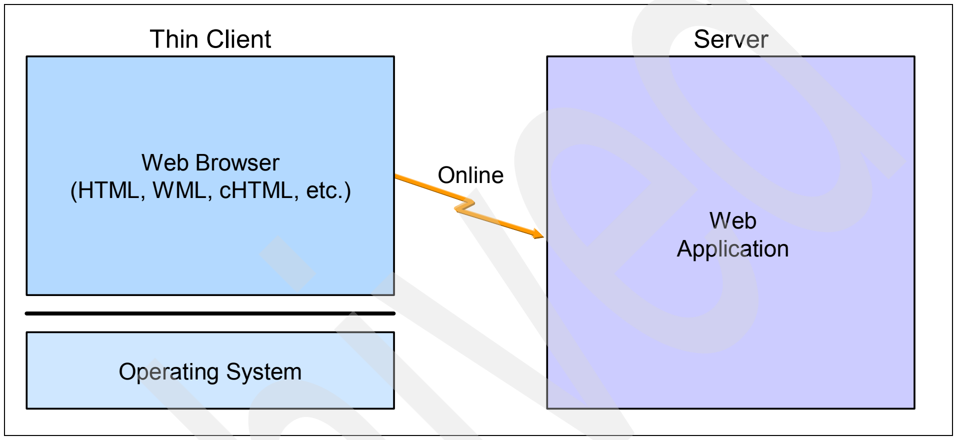


Figure 2-6 Thin Client

Voice-Enabled Client (VEC)

The Voice-Enabled Client has a voice-capable user interface. This includes a simple phone or any IP-based voice client such as a laptop or PDA with microphone and speakers.

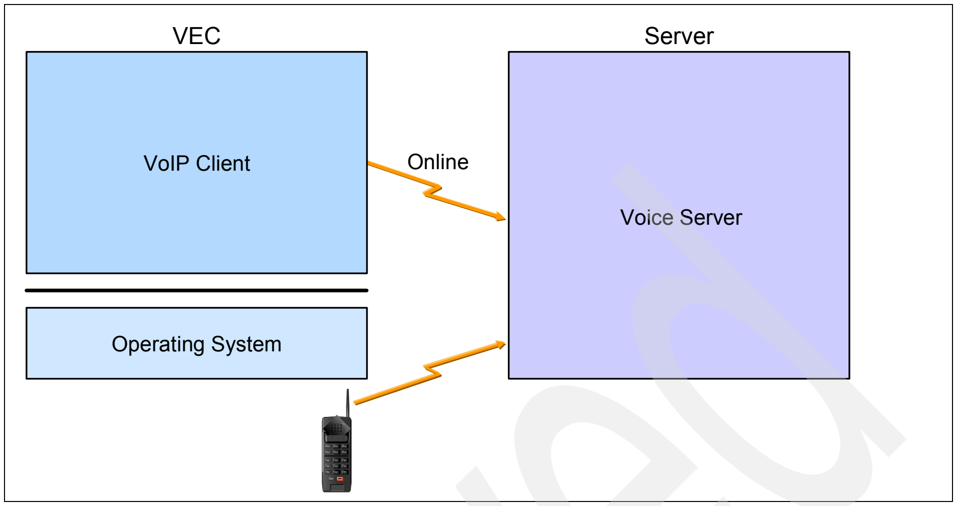


Figure 2-7 Voice-Enabled Client

Distributed Presentation Client (DPC)

A Distributed Presentation Client includes the ability to access online Web content in a Web browser, with the capability of caching the Web content and forms on a mobile device for offline access.

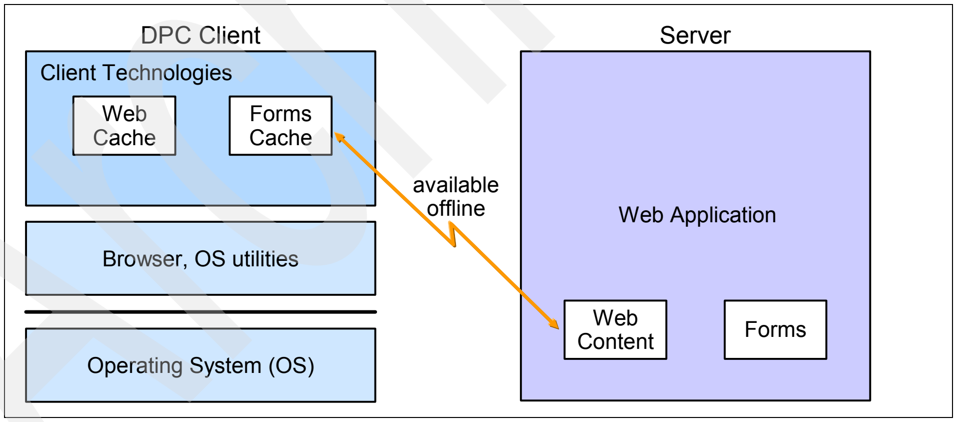


Figure 2-8 Distributed Presentation Client

Distributed Application Client (DAC)

A Distributed Application Client is used for headless applications on a mobile device (for example, embedded device). A device management agent or security agent is commonly used to manage and control the application.

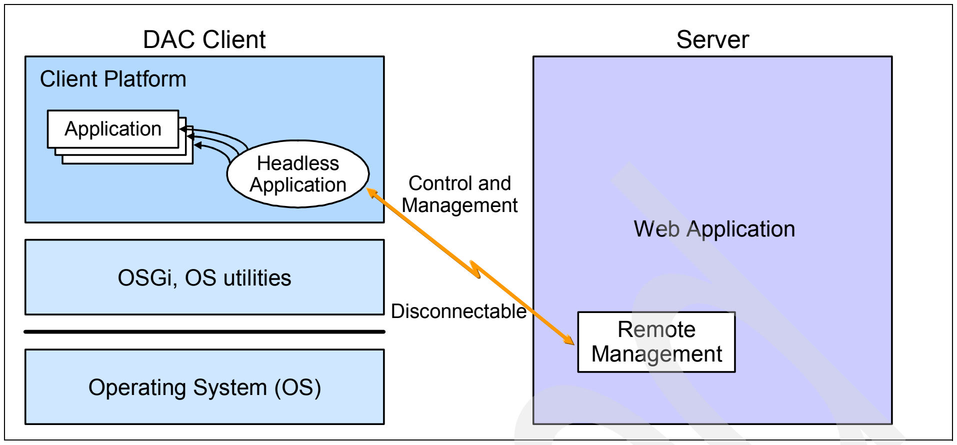


Figure 2-9 Distributed Application Client

Distributed Rich Client (DRC)

The Distributed Rich Client represents a client device that has been enhanced by the installation of rich client software, which gives the device the ability to run in a disconnected and online mode. The user interface (UI) can either be Web UI or rich Eclipse base UI (or direct AWT/SWT for non PC form factor devices). The Distributed Rich Client requires a full or high function device.

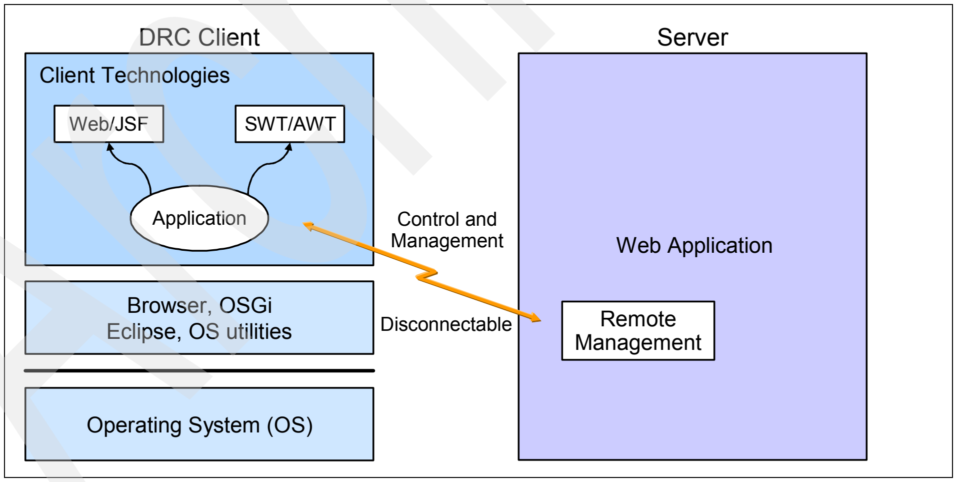


Figure 2-10 Distributed Rich Client

Distributed Collaboration Client (DCC)

The Distributed Collaboration Client provides a collection of typical collaboration functions such as online and offline PIM/e-mail access, instant messaging, people awareness, and office functionality. Some capabilities of the Distributed Collaboration Client can run in disconnected mode, whereas others only work while connected (for example, instant messaging).

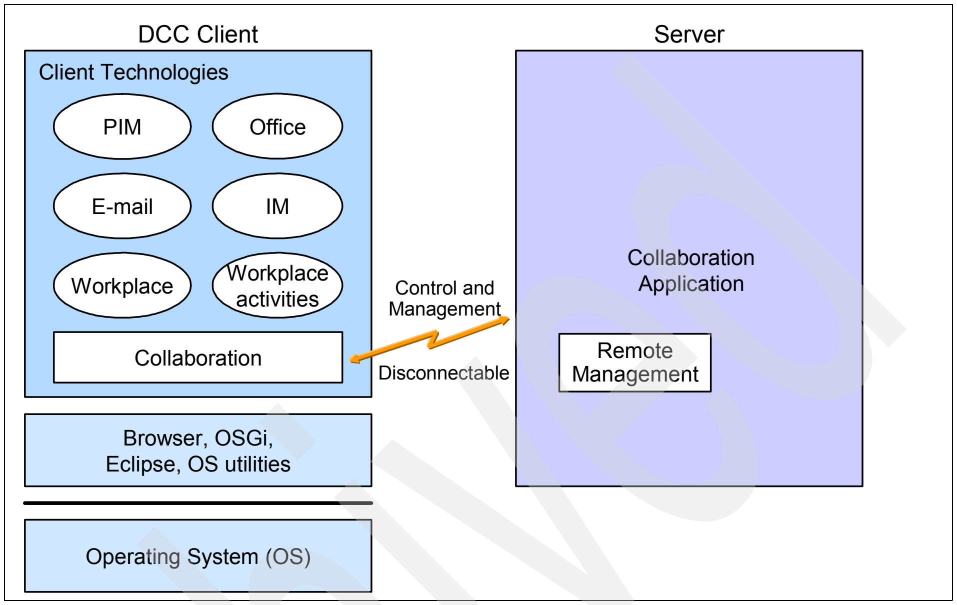


Figure 2-11 Distributed Collaboration Client

Distributed Multimodal Client (DMC)

The Distributed Multimodal Client provides multimodal access services. Multimodal access is the ability to combine multiple modes or channels in the same interaction or session. The Distributed Multimodal Client can run disconnected and in online mode. A common example is a voice and touchscreen controlled navigation system that lets the user use either input type. As a result, the navigation system presents the information as speech and displays the route on the screen.

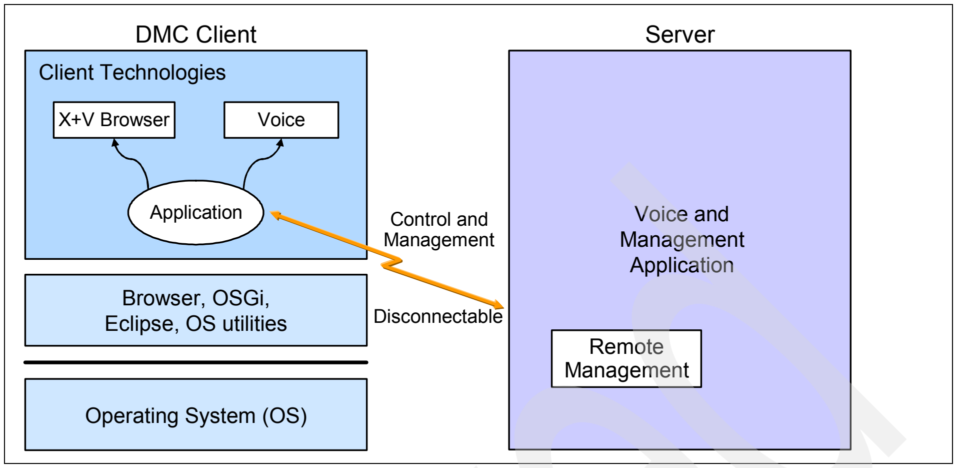


Figure 2-12 Distributed Multimodal Client

Built-in Client (BIC)

The Built-in Client is a generic form of all clients above. It addresses the fact that certain functions are provided by the device manufacturer as built-in applications. The application programmer does not need to create an application for the device anymore. Depending on the built-in function, this client evolves into one of the other clients previously mentioned. For example, if a PIM/e-mail client is built-in, this client is the same as the Distributed Collaboration Client.

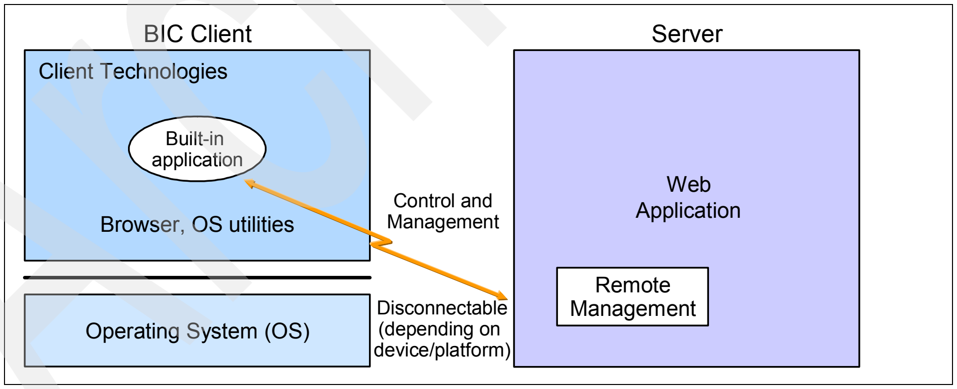


Figure 2-13 Built-in Client

2.3 OSGi

OSGi is a production-ready Java environment for running and managing systems and application bundles across applications in a single JVM. The OSGi model is instantiated in IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (WED client) and IBM WebSphere Everyplace Deployment V6.0 (WED server), as well as other IBM products such as IBM Workplace Client Technology™, Micro Edition (WCTME).

The following quote is from the OSGi Service Platform specification, Release 3 which is found in the file r3.book.pdf in the toolkit product help and also at this Web site:

`http://www.osgi.org`

OSGi was founded in March 1999. Its mission is to create open specifications for the network delivery of managed services to local networks and devices. The OSGi organization is the leading standard for next-generation Internet services to homes, cars, small offices, and other environments.

The OSGi service platform specification delivers an open, common architecture for service providers, developers, software vendors, gateway operators, and equipment vendors to develop, deploy, and manage services in a coordinated fashion. It enables an entirely new category of smart devices due to its flexible and managed deployment of services. The primary targets for the OSGi specifications are set-top boxes, service gateways, cable modems, consumer electronics, PCs, industrial computers, cars, and more. These devices that implement the OSGi specifications enable service providers like telcos, cable operators, utilities, and others to deliver differentiated and valuable services over their networks.

The IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (WED client) framework is built upon the Eclipse Rich Client Platform and OSGi. The OSGi specification enables the separation of the service interface from the service implementation, allowing scalability and extensibility. The WED client framework uses the information in the bundle manifests to populate its service registry and to manage and resolve bundle dependencies. Each bundle has its own class loader and name space, which provides the ability to reference services between bundles within the managed framework.

Bundles are JAR files that include the program code and classes, plus a manifest file with entries specific to the WED client environment, that describe the bundle and its dependencies. The bundle is the program unit whose life cycle is managed by the framework or platform. Bundles can themselves expose further services and Java code that applications (browsers, Web service clients, open sockets) and other bundles can access to perform other tasks such as an HTTP server, Web application, DB2e database, MQe messaging, logging, XML parsing and so on.

The IBM WebSphere Everyplace Client Toolkit V6.0 provides the tools necessary to create and test OSGi bundles, Web Applications, and Embedded Transaction Applications for use on theWebSphere Everyplace Deployment platform. The WebSphere Everyplace Client Toolkit provides wizards that enable developers to create a Client Services project to develop client applications, without requiring them to become OSGi bundle internal experts.

In summary, the OSGi platform is designed to enable devices of any shape and size to execute a consistent, modular program model, on a well-architected set of frameworks and core services. This framework provides services and bundle life cycle management to enable dynamic loading, starting and stopping, and more importantly, bundle and service sharing in the VM instance that is running the WED client platform.

For more information about OSGi, refer to the OSGi home page:

`http://www.osgi.org`

or

`http://www.osgi.org/osgi_technology/index.asp?section=2`

2.4 Server technologies

The services that the client devices connect to reside on a server. Here we discuss the many options available on the server side.

2.4.1 Services

The following technologies are used to host and serve content to mobile devices and manage data exchange with mobile devices.

Application server

An application server allows devices to access and use the applications and other back-end resources managed by the application server. In a mobile environment, an application server may host Web applications that allow devices to access information from servlets or portlets, or it might be used to provide mobile devices with access to services such as device management or synchronization.

Device management

Device management allows you to distribute software to mobile devices, update software on mobile devices, and collect inventory information such as application, hardware, and configuration information in a centralized manner.

Notification services

Notification services allow you to deliver notifications to users based on user preferences and subscriptions. These can be sent via a delivery channel such as SMS, e-mail, or instant messaging. Notifications can also be configured to alert certain groups of users to certain events. For example, server administrators can be alerted when a database server goes down, or financial brokers can be alerted when a stock falls below a certain price.

Synchronization

Synchronization allows the user to maintain accurate copies of data on both the server and on her device. For instance, synchronization of e-mail ensures that the most recent messages display on the device, and synchronization of calendar information ensures that a user’s most recent appointments are viewable on both the device and the PC. Synchronization may also extend to replicating databases or processing message queues between the device and a server.

Content adaptation

Content adaptation (also referred to as transcoding) allows standard HTML-based Web content to be transformed into a size and markup language appropriate for the type of mobile device trying to access this content.

Location Aware Services

Applications using the Location Aware Services are taking advantage of the location of the client device. Based on the location, applications have additional input data to combine into the processing.

Messaging services

Messaging services include the following:

- Transactional support
- Supports a wide range of devices with small, customizable footprint
- Authentication, encryption, non-repudiation, and compression
- Automatic channel management
- Built-in comprehensive security features
- Object messaging (data and function)
- Once-only assured delivery of messages

Web services

Web services provide a standard means of communication among different software applications. With the use of simple foundation technologies used in enabling Web services, it is very simple to create a Web service regardless of the platform, operating system, language, or technology used to implement it.

A *service provider* creates a Web service and publishes its interface and access information to a *service registry* (or *service broker*). A *service requestor* locates entries in the *service registry*, then binds to the *service provider* in order to invoke its Web service.

Web services use the following standards:

- SOAP: An XML-based protocol that defines the messaging between objects
- Web Services Description Language (WSDL): Describes Web services interfaces and access information
- Universal Description, Discovery, and Integration (UDDI): A standard interface for service registries, which allows an application to find organizations and services

Voice services

Special *voice servers* provide voice services for the system, including:

- Automatic Speech Recognition (ASR)
- Text-to-Speech (TTS)
- Natural Language Understanding (NLU)

2.4.2 Java-based technologies

Because Java allows developers to write an application once and run it anywhere, Java-based technologies are very significant in the area of pervasive computing where there are many varied devices on which to run the same or similar applications.

J2SE

Java 2 Platform Standard Edition (J2SE) is an edition of Sun™'s Java platform, designed to allow applications to be written once and run anywhere. It defines the Java Virtual Machine and core class libraries, and it provides the foundation for building applications, browser-based applets, and other Web applications.

For more information about J2SE and the following Java-based technologies, refer to Sun's Web site:

http://java.sun.com

J2EE

Java 2 Platform Enterprise Edition (J2EE) is a super set of J2SE that enables the development of robust and complex server-based enterprise applications. J2EE provides such technology components as Enterprise JavaBeans (EJBs) and Web application services.

Servlets

Servlets are Java-based software components that can respond to HTTP requests with dynamically generated HTML. Servlets are more efficient than CGI for Web request processing since they do not create a new process for each request.

Servlets run within a Web container as defined by the J2EE Model and therefore have access to the rich set of Java-based APIs and services.

One of the attractions of using servlets is that the API is a very accessible one for a Java programmer to master.

JavaServer Pages

JSPs provide a server-side scripting technology that enables Java code to be embedded within Web pages, so JSPs have the appearance of HTML pages with embedded Java code. When the page is executed, the Java code can generate dynamic content to appear in the resulting Web page. JSPs are compiled at runtime into servlets that execute to generate the resulting HTML. Subsequent calls to the same JSP simply execute the compiled servlet.

JSP technology uses XML-like tags and scriptlets written in Java programming language to encapsulate the conditional logic that generates dynamic content for an HTML page. In the runtime environment, JSPs are compiled into servlets before being executed on the Web application. Output is not limited to HTML but also includes WML, XML, cHTML, DHTML, and VoiceXML.

JSPs are the recommended choice for implementing the view that is sent back to the Web client. For those cases where the code required on the page is to be a large percentage of the page, and the HTML minimal, writing a Java servlet makes the Java code much easier to read and therefore maintain.

JavaBeans

JavaBeans are an architecture developed by Sun Microsystems™ describing an API and a set of conventions for reusable, Java-based components. Code written to Sun's JavaBeans architecture is called JavaBeans or just beans.

Beans are recommended for use in conjunction with servlets and JSPs in the following ways:

- As the client interface to the Model layer. An Interaction Controller servlet uses this bean interface.
- As the client interface to other resources. In some cases this may be generated for you by a tool.
- As a component that incorporates a number of property-value pairs for use by other components or classes. For example, the JavaServer™ Pages™ specification includes a set of tags for accessing JavaBeans properties.

Enterprise JavaBeans

Enterprise JavaBeans is Sun's trademarked term for its EJB architecture (or *component model*). When writing to the EJB specification, you are developing *enterprise beans* (or, if you prefer, EJBs).

Enterprise beans are distinguished from JavaBeans in that they are designed to be installed on a server and accessed remotely by a client. The EJB framework provides a standard for server-side components with transactional characteristics. The EJB developer specifies the required transactional and security characteristics of an EJB in a deployment descriptor (this is sometimes referred to as declarative programming). In a separate step, the EJB is then deployed to the EJB container provided by the application server vendor of your choice.

There are three types of Enterprise JavaBeans:

- Session beans
- Entity beans
- Message-driven beans

Java Message Service (JMS)

Java Message Service is an API that allows developers to create Java-based applications that can exchange data with an enterprise messaging system in a standardized way.

JMS defines a set of common enterprise messaging concepts in order to maximize portability of the same code base among different enterprise messaging systems.

For more information about JMS, see Sun’s Web site:

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

Portlets

Portlets are reusable components that provide access to Web-based contents, applications, and other resources. Web pages, Web Services, applications, and syndicated content feeds can be accessed through portlets.

Companies can create their own portlets or select portlets from a catalog of third-party portlets. Portlets are intended to be assembled into a larger portal page, with multiple instances of the same portlet displaying different data for each user.

From a user's perspective, a portlet is a window on a portal site that provides a specific service or information, for example, a calendar or news feed. From an application development perspective, portlets are pluggable modules that are designed to run inside a portlet container of a portal server.

Portlets are coded against the portlet API. The portlet components are part of the portal application. The portal application is deployed on the portal server.

2.5 Connectivity technologies

There are also many options to connect the client devices to their server counterparts. In a mobile environment, wireless technologies are the ones most often considered.

2.5.1 Wireless technologies

In a mobile environment, wireless connectivity technologies are the most likely to be used. When choosing which wireless connectivity technology, it is important to factor speed, cost, and coverage into your decision.

Cellular

Cellular networks provide the greatest areas of coverage. There are many different cellular technologies that allow Web access to wireless devices such as GPRS, GSM, CDMA, and UMTS. Because this area is always changing and evolving, it is hard to give exact specifications on cellular coverage currently offered by wireless providers. Current speeds are near that of a 56-k modem. In the near future, these speeds should approach near-broadband speed in metropolitan areas.

Wireless Ethernet

Wireless Ethernet (802.11x) is the most popular technology for providing wireless access in buildings, shops, homes, and workplaces. It is faster than cellular, but not as fast as wired Ethernet. It is reliable and can be easily added to an existing wired network infrastructure. Speeds range from 11mbps to 54 mbps but can degrade the farther a user gets from a wireless access point.

Wireless Ethernet standards such as 802.16 are currently under development for wide-scale implementations of wireless Ethernet in the form of a Metropolitan Area Network.

More information about wireless Ethernet can be found at:

http://www.ieee802.org

Bluetooth and Infrared (IR)

Bluetooth is another wireless standard used to allow devices within relatively close proximity to exchange data.

For more information, see the Bluetooth Web site:

http://www.bluetooth.org

Infrared uses a beam of infrared light to exchange data in the same way a remote control is used to change a television channel. It is slower than Bluetooth, and the devices must be positioned in direct line-of-sight from each other to communicate.

2.5.2 Wired technologies

Below is a list of wired technologies used in today's networks.

Plain Old Telephony Services (POTS)

Telephony services can be used to connect to IT networks (for example, a modem connection from a desktop). Even though this is an old technology, it is still a valid and working method today.

Digital Subscriber Line (DSL), Cable

On digital networks data can be transmitted with a higher rate. DSL is a a successor of the analog phone network, and makes a digital connection between subscriber and provider. Cable networks are based on the digital Cable TV networks.

Cradle

Mobile devices such as PDAs come with cradles that connect to a PC to synchronize data. Often, this same connection can enable the device to access other network resources and the Internet. For situations where wireless connectivity is not an option, the cradle provides an opportunity for the user to synchronize data with a network server or PC.

Ethernet

Some devices can be connected to a network via the Ethernet in the same way that many PC workstations are. The Ethernet provides relatively inexpensive connectivity, easy integration with an already existing wired network, and high speeds.

2.5.3 Issues with connectivity

When implementing a mobile environment, you must give due consideration for:

- Network security - Who is authorized to use your services, how are they authenticated, and if your data needs to be encrypted.
- Connectivity management - Will users roam between different wireless and wired networks?
- Bandwidth - How much bandwidth will be used and how expensive will this be?

2.6 Mobile Web technologies

There are many technologies that relate to the area of the mobile Web. Some of these are markup languages. Others play a role in user interaction or controlling how data is exchanged with mobile devices.

2.6.1 HTML

HyperText Markup Language (HTML) is a document markup language with support for hyperlinks that is rendered by the browser. HTML uses tags to structure text into headings, paragraphs, lists, hypertext links, and so forth. Many e-business applications are assembled strictly using HTML.

This has the advantage that the client-side Web application can be a simple HTML browser, enabling a less capable client to execute an e-business application.

2.6.2 cHTML

cHTML stands for Compact HTML and is a subset of the HTML specifications targeting small appliances such as smartphones and mobile PDAs. The cHTML tries to bypass several hardware restrictions by providing a standard markup language and small browser that could be executed in a constrained environment with a small memory, low power CPU, a small display, etc.

Since the Compact HTML is based on standard HTML recommendations from W3C, we can develop and apply software tools to adapt pure HTML to cHTML, making Internet information available and adequately formatted to new classes of devices and appliances. Basically, cHTML excludes JPEG images, tables, image maps, multiple character fonts and styles, background color or images, frames, and cascading style sheets from the HTML specification.

cHTML is the markup language of i-Mode. i-Mode is a wireless service developed by NTT DoCoMo in Japan. It is designed to provide mobile phone voice service, and Internet and e-mail access. For more information about Compact HTML, you can read the document submitted to W3C, World Wide Web Consortium, at:

`http://www.w3.org/TR/1998/NOTE-compactHTML-19980209`

2.6.3 XML

XML allows you to specify your own markup language with tags specified in a Document Type Definition (DTD) or XML Schema. Actual content streams are then produced that use this markup. The content streams can be transformed to other content streams by using Extensible Stylesheet Language (XSL), which is based on CSS.

For PC-based browsers, HTML is well established for both document content and formatting. The leading browsers have significant investments in rendering engines based on HTML and a Document Object Model (DOM) based on HTML for manipulation by JavaScript™.

XML seems to be evolving to a complementary role for active content within HTML documents for the PC browser environment.

For new devices, such as WAP-enabled phones and voice clients, the data content and formatting is being defined by new XML schema, WML for WAP phone, and VoiceXML for voice interfaces.

For most Web application designs, you should focus your attention on the use of XML on the server side.

2.6.4 XML Device-Independent Markup Extensions (XDIME)

XDIME is a markup language that attempts to enable developers to code an application once and run it on any device.

2.6.5 XForms

XForms is W3C's specification for Web forms that can be used with desktop computers, hand-held devices, etc. The disadvantage of the HTML Web forms is that there is no separation of purpose from presentation. XForms separate the data and logic of a form from its presentation. Also, XForms are device-independent.

XForms use XML for transporting the data that is displayed on the form and the data that is submitted from the form. HTML is used for the data display. For more information about XForms, see:

`http://www.w3.org/TR/xforms`

2.6.6 XHTML 1.1 (HTML 4.01)

Extended HyperText Markup Language (XHTML) is an extension to HTML 4, which supports document types that are XML-based. It is intended to be used as a language for XML-conforming content as well as for HTML 4-conforming user agents.

The advantages of XHTML are as follows:

- Since XHTML documents are XML conforming, they can be viewed, edited, and validated with standard XML tools.
- XHTML documents can be used to traverse either the HTML Document Object Model or the XML Document Object Model.

Some issues with XHTML are:

- XHTML documents are not as easy to create as HTML documents because XHTML is validated more strictly than HTML.
- HTML is already used so widely that it is difficult for XHTML to attract the attention of most Web developers.
- Browser support is not usually an issue since documents can be created using HTML-compatible XHTML that is understood by most browsers. There are also utilities that can be used to convert HTML documents to HTML-compatible XHTML.
- Development tool support for XHTML is also improving. The Page Designer tool in IBM Rational Application Developer V6, for example, allows visual authoring of XHTML pages.

XHTML Basic is designed for Web clients that do not support the full set of XHTML features. It is meant to serve as a common language and share basic content across mobile phones, pagers, car navigation systems, vending machines, etc.

Some of the common features found in Wireless Markup Language (WML) and other subsets of HTML have been used as the basis for developing XHTML.

Basic:

- Basic text
- Basic forms and tables
- Hyperlinks

Some HTML 4 features have been found inappropriate for non-desktop devices, so extending and building on XHTML Basic helps bridge that gap.

2.6.7 XSLT

Extensible Stylesheet Language Transformations (XSLT) is a W3C specification for transforming XML documents into other documents, including other XML documents, HTML documents, and WML documents. The XSLT is built on top of the Extensible Stylesheet Language (XSL), a style sheet language for XML (such as CSS2 for HTML). Unlike CSS2, XSL is also a transformation language.

A transformation expressed in the XSLT language defines a set of rules for transforming a source tree to a result tree, and it is expressed in the form of a style sheet.

2.6.8 WML

The Wireless Markup Language (WML) is based on XML and HTML 4.0 to fit small hand-held devices. It is a tag-based language that handles formatting static text and images, can accept data input, and can follow hyperlinks. WML also uses WMLScript, a compact JavaScript-like language that runs in limited memory. WML is the markup language of WAP.

The WML specification is maintained by The Open Mobile Alliance. The Open Mobile Alliance has been established by the consolidation of the WAP Forum and the Open Mobile Architecture Initiative, two industry-wide consortiums concerned with the development of an open standard for the wireless industry.

For more information you can visit The Open Mobile Alliance Web site at:

`http://www.openmobilealliance.org`

or

`http://www.wapforum.org`

2.6.9 SyncML DS and DM

SyncML stands for Synchronization Markup Language. It is a an open standard protocol that comes in two flavors: One for data synchronization and one for device management.

The goal of SyncML DS is to enable synchronization of any type of data, from any application, on any device, and over any network. It has been designed to cope well with the specificities of mobile phones such as low bandwidth, unreliable connections, and high network latency.

SyncML DM is based on the SyncML protocol and is designed to enable the customizing, personalization, and servicing of mobile device, taking into account the same limitations as SyncML DS does for mobile environments.

For more information you can visit the SyncML Official Web Site at:

http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html

2.6.10 VoiceXML and X+V

The Voice eXtensible Markup Language (VoiceXML) is an XML-based industry standard language for creating voice applications, much as HTML is a language for developing visual applications.

VoiceXML is defined and promoted by an industry forum, the VoiceXML Forum, founded by AT&T, IBM, Lucent, and Motorola, and currently supported by more than 570 member companies.

VoiceXML was designed to create audio dialogs that feature text-to-speech, digitized as well as prerecorded audio, recognition of both spoken and dual-tone multi-frequency (DTMF) key input, recording of spoken input, telephony, and mixed-initiative conversations. Its goal is to provide voice access to Web-based content and applications. It enables the development of voice applications via the use of a familiar markup style and Web server-side logic to deliver applications over telephone lines. The resulting applications allow conversational access to Web-based data, and can also interact with existing back-end business data and logic.

A VoiceXML application is capable of retrieving information from a Web server and, by making use of scripts and appropriate grammars, the application can interact with the customer through spoken words.

For more information you can visit the VoiceXML Forum Official Web site at:

http://www.voicexml.org

X+V

X+V is an abbreviation of XHTML + VoiceXML, and is a markup language specification submitted to the World Wide Web Consortium (W3C) by IBM, Motorola, and Opera Software to simplify the development of multimodal applications.

Multi-modal access gives users of pervasive devices (smart phones, PDAs, kiosks, set-top-boxes, etc.) a range of options for interacting with an application. To input information, they might use some combination of voice, keypad, stylus, touchscreen, and the application would deliver information using a combination of speech synthesis, text, graphics, A/V, etc.

X+V allows you to operate in a voice-only environment, in a visual-only environment, and (if you want) in a multimodal environment.

Part 2

SOA Client - Access Integration patterns

Part 2 guides the reader through the process of reusing the patterns assets within the context of the Rational Unified Process (RUP) to accelerate the solution design. We include selection criteria to navigate from the Business and IT drivers to the appropriate Application pattern or patterns, and supporting Runtime pattern, component model, and Product mapping instantiated by IBM software products.

3 Design a solution using the Patterns for e-business

The role of an IT architect is to evaluate business problems and build solutions to solve them. The architect begins by gathering input on the problem, developing an outline for the desired solution, and considering any special requirements that need to be factored into that solution. The architect then takes this input and designs the solution, which can include one or more computer applications that address the business problems by supplying the necessary business functions.

To improve the process over time, we need to capture and reuse the experience of the IT architects in such a way that future engagements can be made simpler and faster. We do this by capturing knowledge gained from each engagement and using it to build a repository of assets. IT architects can then build future solutions based on these proven assets. This reuse saves time, money, and effort, and helps ensure delivery of a solid, properly architected solution.

The IBM Patterns for e-business help facilitate this reuse of assets. Their purpose is to capture and publish e-business artifacts that have been used, tested, and proven to be successful. The information captured by them is assumed to fit the majority, or 80/20, situation. The IBM Patterns for e-business are further augmented with guidelines and related links.

The objective of this chapter is to describe how the Patterns for e-business can be used within the big picture context of a development methodology such as the

Rational Unified Process (RUP) or IBM Method. In addition, we provide an explanation of the terminology and syntax used to describe the Patterns for e-business.

Note: For more information about patterns with RUP refer to *Using a Single Business pattern with Rational Unified Process*, REDP-3812.

The chapter is organized into the following sections:

- Applying patterns within the development life cycle
- The Patterns for e-business layered asset model
- How to use the Patterns for e-business
- Patterns for e-business naming conventions
- Summary

3.1 Applying patterns within the development life cycle

The objective of this section is to describe to IT architects, IT specialists, and developers how the IBM Patterns for e-business can be used within the context of a life-cycle methodology. For illustration purposes, we use the Rational Unified Process (RUP); however, the concepts can be applied to other methodologies such as IBM Method.

Note: If you are not familiar with the Rational Unified Process, we recommend you review Appendix C, "Rational Unified Process (RUP) overview" on page 483.

Figure 3-1 depicts the RUP disciplines for each project phase. As seen in Figure 3-1, the IT solution architect provides their greatest effort in the *Inception and the Elaboration phases*. The Patterns for e-business are used primarily to accelerate the architecture development steps found in the RUP *Requirements and Analysis and Design disciplines*.

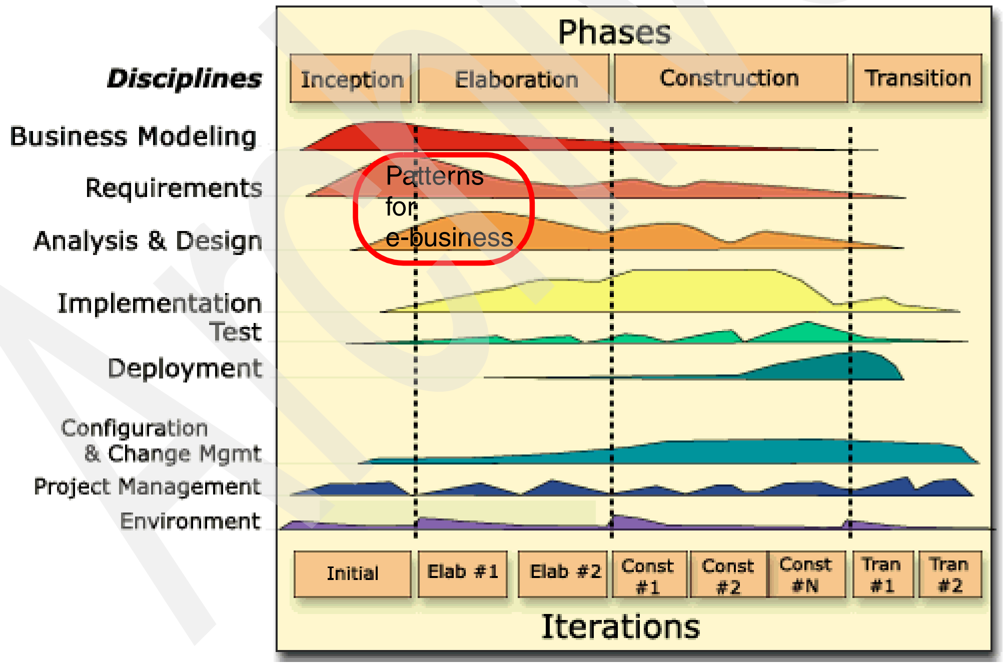
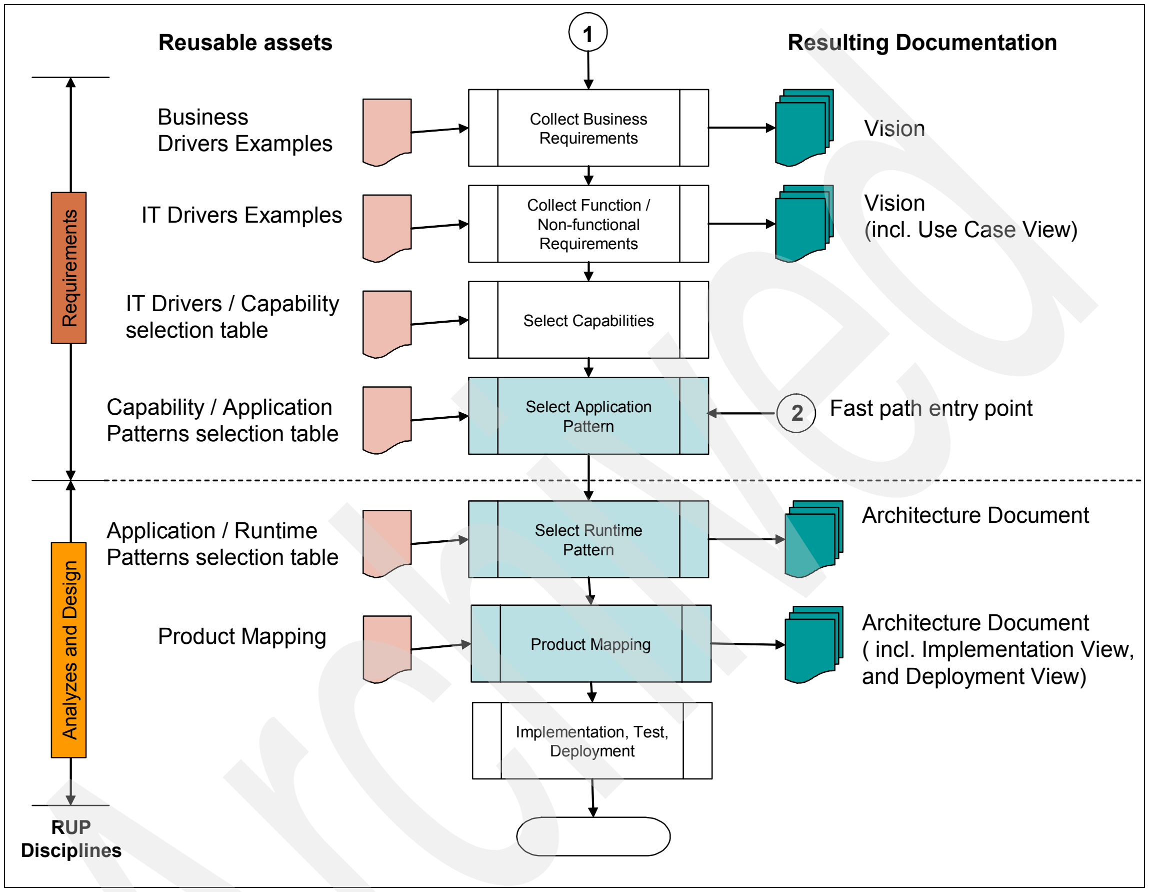


Figure 3-1 Architecture of RUP

3.1.1 Process for using patterns within RUP

Figure 3-2 depicts the process for using the reusable assets found in this book within the context of RUP (Requirements, and Analysis and Design disciplines).



Reusable assets
1
Resulting Documentation
Business Drivers Examples
Collect Business Requirements
Vision
IT Drivers Examples
Collect Function / Non-functional Requirements
Vision (incl. Use Case View)
Requirements
IT Drivers / Capability selection table
Select Capabilities
Capability / Application Patterns selection table
Select Application Pattern
2
Fast path entry point
Application / Runtime Patterns selection table
Select Runtime Pattern
Architecture Document
Analyzes and Design
Product Mapping
Product Mapping
Architecture Document (incl. Implementation View, and Deployment View)
Implementation, Test, Deployment
RUP Disciplines

Figure 3-2 Process for using the patterns within RUP

There are a couple of key points in the process displayed in Figure 3-2 that we would like to highlight:

- The *Reusable assets* are the patterns assets documented in this book that are used to accelerate the activities and resulting output.
- The blue boxes (Select Application Pattern, Select Runtime Pattern, Product mapping) are specific to the patterns, whereas the white boxes are defined as part of the methodology.

- Depending on your knowledge level, there are different entry points for navigating the process to the point of selecting the Application pattern or patterns.
  - *Entry point 1:* This entry point is appropriate for architects less familiar with the Access Integration patterns. This entry point provides selection criteria and decision support when capturing IT requirements and mapping the requirements to capabilities, which are used to select Application patterns.
  - *Entry point 2 (Fast path):* This entry point is appropriate for architects that are familiar with the Access Integration pattern and already understand the meaning of the capabilities. In this case, the architect maps the customer IT requirements to the capabilities used to select the Application pattern from their knowledge of the Application patterns.

The high-level process for using the patterns within RUP is as follows:

1. Collect business requirements.

   Refer to 4.2, “Business drivers” on page 85, as inspiration and input for the discussion with the client and the creation of the *Business requirements document* (vision). SOA user and device access solutions typically fall into the Access Integration pattern.

2. Collect functional and non-functional IT requirements.

   Refer to 4.3, “IT drivers” on page 87, as input for the discussion with the client. Identify which IT drivers apply and document them in the vision document and in the *Use Case View*.

3. Select capabilities based on the IT requirements and use cases.

   Refer to 4.5, “Select capabilities based on IT drivers” on page 95, to identify the capabilities the solution must provide.

4. Select the matching Application pattern or patterns.

   Refer to 5.2, “Select the Application pattern” on page 103, to determine the appropriate application pattern or patterns based on the capabilities needed to fulfill the requirements.

5. Select the matching Runtime pattern or patterns.

   Each Application pattern will reference a Runtime pattern.

6. Select products (Product mapping).

   A Product mapping based on the IBM pervasive software is shown to fulfill the function of each of the logical nodes of the Runtime environment.

   A component model is created in the *Implementation View*. The *Deployment View* is based on the chosen products (also referred to as Operational Model).

7. Implement, test, and deploy.

Important: Part 3, “ITSO MobileAdjuster patterns working example” on page 235, walks the reader through the end-to-end development process to demonstrate how the Access Integration pattern can be used within the context of RUP, to accelerate the solution architecture and design.

3.1.2 Mapping reusable patterns assets to RUP activities

Table 3-1 shows the RUP activities, and the corresponding tasks and artifacts that support the IT Architect during the solution architecture development. The objective of this table is to highlight where the Patterns for e-business can be used within RUP, and provide working example chapters that demonstrate the use of the patterns.

Table 3-1 Patterns for e-Business and RUP disciplines, activities, and artifacts relation in this book

RUP disciplines	Input	Key activities	Output	Patterns chapters	Working example chapters
Business modeling (optional)	Business process analyzes	Explore process automation	Input for Vision artifact		
Requirements Activity: Develop Vision (Vision contains functional and non-functional requirements, Use Case View)	* Business requirements template	* Collect business requirements	Business requirements document	4.2, “Business drivers” on page 85	8.1, “Business modeling” on page 198
	* Business requirements document * IT drivers template	* Collect functional non-functional IT requirements	* IT requirements document	4.3, “IT drivers” on page 87	* 8.2, “IT requirements” on page 203
	* IT requirements document	Model use cases	* Use Case View		8.3, “Use cases” on page 204
	* IT requirements * Use case view	Select capabilities based on IT requirements	* Capabilities	* 4.5, “Select capabilities based on IT drivers” on page 95	* 8.4.1, “Select capabilities based on requirements” on page 220

RUP disciplines	Input	Key activities	Output	Patterns chapters	Working example chapters
Analysis and Design Activity: Architectural Analysis, Define and Refine Architecture	* List of capabilities	* Select Business, Composite, Integration patterns * Select the application pattern or patterns	* Application pattern or patterns	5.2, “Select the Application pattern” on page 103	* 8.4.2, “Select the Application pattern” on page 224
	* List of application patterns	* Select the runtime pattern or patterns	* Runtime pattern or patterns	6.2, “Select Runtime pattern and Product mapping” on page 134	* 8.4.3, “Select the Runtime pattern and Product mapping” on page 226
	* List of runtime patterns * Component model	* Map products * Create Implemen-tation View (component model)	* Product mapping * Implemen-tation View (component model)	6.2, “Select Runtime pattern and Product mapping” on page 134	* 8.4.3, “Select the Runtime pattern and Product mapping” on page 226
	* Product mapping * Implemen-tation view	* Create deployment view	Deployment view		
Implementation					Chapter 11, “Application development” on page 279
Test					* Chapter 11, “Application development” on page 279 * Chapter 15, “Application walkthrough” on page 407

RUP disciplines	Input	Key activities	Output	Patterns chapters	Working example chapters
Deployment					* Chapter 13, “Application deployment” on page 345 * Chapter 15, “Application walkthrough” on page 407
Configuration and Change Management					
Project Management					
Environment					* Appendix C, “Rational Unified Process (RUP) overview” on page 483 * Chapter 10, “Development environment installation” on page 255

3.2 The Patterns for e-business layered asset model

The Patterns for e-business approach enables architects to implement successful e-business solutions through the reuse of components and solution elements from proven successful experiences. The Patterns approach is based on a set of layered assets that can be exploited by any existing development methodology. These layered assets are structured in a way that each level of detail builds on the last and include:

- Business patterns that identify the interaction between users, businesses, and data
- Integration patterns that tie multiple Business patterns together when a solution cannot be provided based on a single Business pattern
- Composite patterns that represent commonly occurring combinations of Business patterns and Integration patterns

- Application patterns that provide a conceptual layout that describes how the application components and data within a Business pattern or Integration pattern interact
- Runtime patterns that define the logical middleware structure that supports an Application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes
- Product mappings that identify proven and tested software implementations for each Runtime pattern
- Best-practice guidelines for design, development, deployment, and management of e-business applications

Figure 3-3 shows these assets and their relationships to each other.

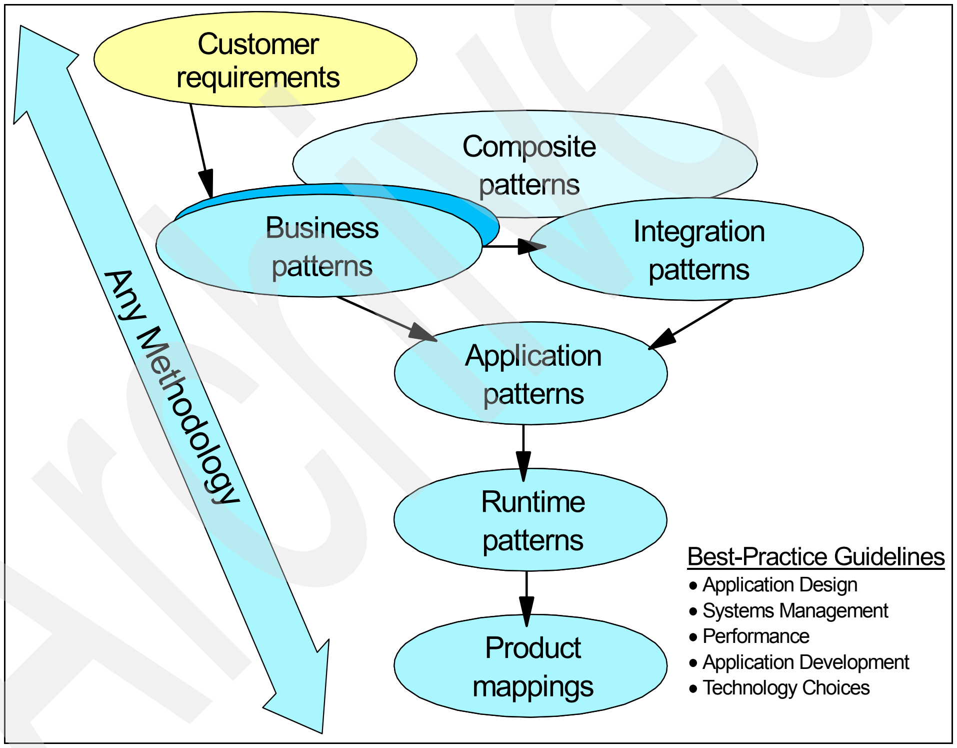


Figure 3-3 The Patterns for e-business layered asset model

Patterns for e-business Web site

The layers of patterns, along with their associated links and guidelines, allow the architect to start with a problem and a vision for the solution and then find a pattern that fits that vision. In order to navigate from top down from one level to another, a decision matrix will be provided to assist the architect in making the right decision.

Then, by drilling down using the patterns process, the architect can further define the additional functional pieces that the application needs to succeed. Finally, the architect can build the application using coding techniques that are outlined in the associated guidelines.

The Patterns Web site provides an easy way to navigate through the layered Patterns assets to determine the most appropriate assets for a particular engagement.

For easy reference, see the Patterns for e-business Web site at:

http://www.ibm.com/developerWorks/patterns/

3.3 How to use the Patterns for e-business

As described in the previous section, the Patterns for e-business have a layered structure where each layer builds detail on the last. At the highest layer are Business patterns. These describe the entities involved in the e-business solution.

Composite patterns appear in the hierarchy shown in Figure 3-3 on page 69 above the Business patterns. However, Composite patterns are made up of a number of individual Business patterns and at least one Integration pattern. This section discusses how to use the layered structure of Patterns for e-business assets.

3.3.1 Selecting a Business, Integration, or Composite pattern, or a Custom design

When faced with the challenge of designing a solution for a business problem, the first step is to get a high-level view of the goals that you are trying to achieve. You need to describe a proposed business scenario and match each element to an appropriate IBM Pattern for e-business. You might find, for example, that the total solution requires multiple Business and Integration patterns or that it fits into a Composite pattern or Custom design.

For example, suppose an insurance company wants to reduce the amount of time and money spent on call centers that handle customer inquiries. By allowing customers to view their policy information and request changes online, the company can cut back significantly on the resources that are spent handling this type of request by phone. The objective allows policy holders to view policy information that is stored in legacy databases.

The Self-Service business pattern fits this scenario perfectly. You can use it in situations where users need direct access to business applications and data. The following sections discuss the available Business patterns.

Business patterns

A Business pattern describes the relationship between the users, the business organizations or applications, and the data to be accessed.

There are four primary Business patterns, which are explained in Table 3-2.

Table 3-2 The four primary Business patterns

Business Patterns	Description	Examples
Self-Service (user-to-business)	Applications where users interact with a business via the Internet or intranet	Simple Web applications
Information Aggregation (user-to-data)	Applications where users can extract useful information from large volumes of data, text, images, and so forth	Business intelligence, knowledge management, and Web crawlers
Collaboration (user-to-user)	Applications where the Internet or intranet supports collaborative work between users	Community, chat, video conferencing, e-mail, and so forth
Extended Enterprise (business-to-business)	Applications that link two or more business processes across separate enterprises	EDI, supply chain management, and so forth

It would be very convenient if all problems fit nicely into these four slots, but reality says that things can often be more complicated. The patterns assume that most problems, when broken down into their basic components, will fit more than one of these patterns. When a problem requires multiple Business patterns, you can use Integration patterns.

Integration patterns

Integration patterns allow you to tie together multiple Business patterns to solve a business problem. Table 3-3 describes the Integration patterns.

Table 3-3 Integration patterns

Integration patterns	Description	Examples
Access Integration	Integration of a number of services through a common entry point	Portals
Application Integration	Integration of multiple applications and data sources without the user directly invoking them	Message brokers, workflow managers, data propagators, and data federation engines

The Access Integration pattern maps to User Integration. The Application Integration pattern is divided into two essentially different approaches:

- Process Integration, which is the integration of the functional flow of processing between the applications
- Data Integration, which is the integration of the information that is used by applications

You can combine the Business and Integration patterns to implement installation-specific business solutions called a Custom design.

Custom design

Figure 3-4 illustrates the use of a Custom design to address a business problem.

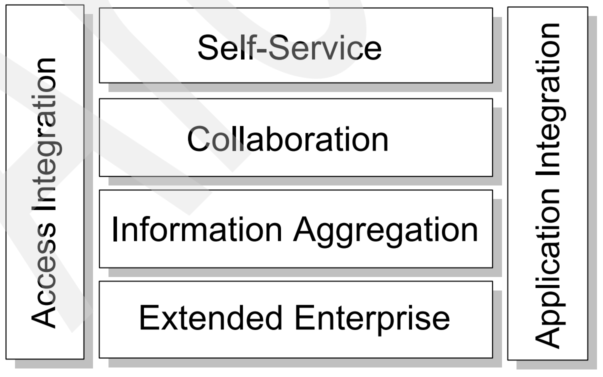


Figure 3-4 Patterns representing a Custom design

If you do not use any of the Business or Integration patterns in a Custom design, you can show the unused patterns as lighter blocks than those patterns that you do use. For example, Figure 3-5 shows a Custom design that does not have a Collaboration or an Extended Enterprise business pattern for a business problem.

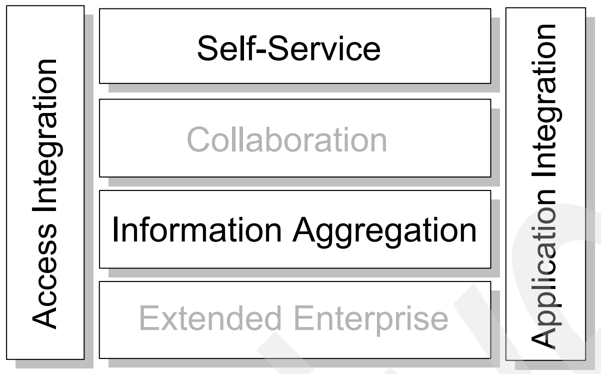


Figure 3-5 Custom design showing unused patterns

If a Custom design recurs many times across domains that have similar business problems, then it can also be a Composite pattern. For example, the Custom design in Figure 3-5 can also describe a Sell-Side Hub Composite pattern.

Composite patterns

Several common uses of Business and Integration patterns have been identified and formalized into Composite patterns. Table 3-4 on page 74 shows the identified Composite patterns.

Table 3-4 Composite patterns

Composite patterns	Description	Examples
Electronic Commerce	User-to-online-buying.	• http://www.macys.com • http://www.amazon.com
Portal	Typically designed to aggregate multiple information sources and applications to provide uniform, seamless, and personalized access for its users.	• Enterprise intranet portal providing self-service functions such as payroll, benefits, and travel expenses. • Collaboration providers who provide services such as e-mail or instant messaging.
Account Access	Provides customers with around-the-clock account access to their account information.	• Online brokerage trading applications. • Telephone company account manager functions. • Bank, credit card, and insurance company online applications.
Trading Exchange	Allows buyers and sellers to trade goods and services on a public site.	• Buyer's side - Interaction between buyer's procurement system and commerce functions of e-Marketplace. • Seller's side - Interaction between the procurement functions of the e-Marketplace and its suppliers.
Sell-Side Hub (supplier)	The seller owns the e-Marketplace and uses it as a vehicle to sell goods and services on the Web.	http://www.carmax.com (car purchase)
Buy-Side Hub (purchaser)	The buyer of the goods owns the e-Marketplace and uses it as a vehicle to leverage the buying or procurement budget in soliciting the best deals for goods and services from prospective sellers across the Web.	http://www.wwre.org (WorldWide Retail Exchange)

The makeup of these patterns is variable in that there will be basic patterns present for each type. However, you can extend the Composite pattern to meet additional criteria. For more information about Composite patterns, refer to *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos.

3.3.2 Selecting Application patterns

After you identify the Business pattern, the next step is to define the high-level logical components that make up the solution and how these components interact. This is known as the Application pattern. A Business pattern usually has multiple possible Application patterns. An Application pattern might have logical components that describe a presentation tier for interacting with users, an application tier, and a back-end application tier.

Application patterns break down the application into the most basic conceptual components that identify the goal of the application. In our example, the application falls into the Self-Service business pattern, and the goal is to build a simple application that allows users to access back-end information. Figure 3-6 shows the Self-Service::Directly Integrated Single Channel application pattern, which fulfills this requirement.

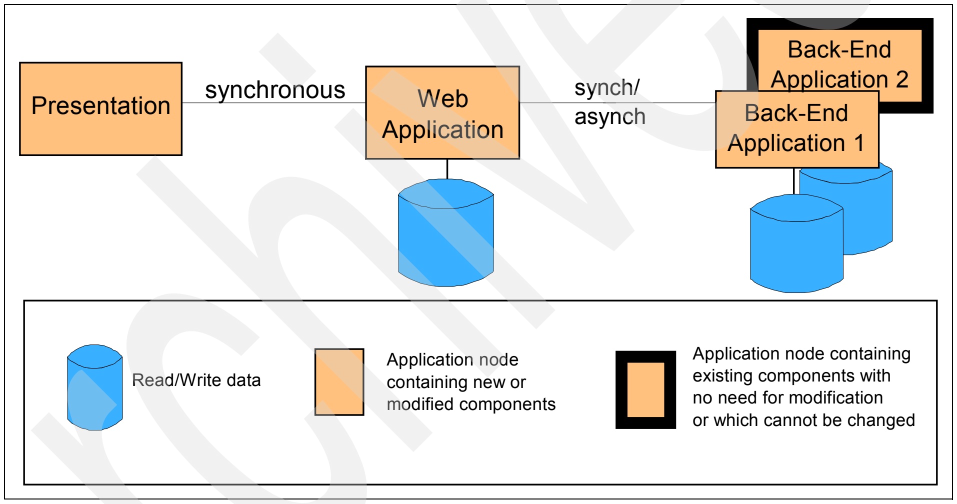

Presentation
synchronous
Web Application
synch/ asynch
Back-End Application 2
Back-End Application 1
Read/Write data
Application node containing new or modified components
Application node containing existing components with no need for modification or which cannot be changed

Figure 3-6 Self-Service::Directly Integrated Single Channel pattern

This Application pattern consists of a presentation tier that handles the request and response to the user. The application tier represents the component that handles access to the back-end applications and data. The multiple application boxes on the right represent the back-end applications that contain the business data. The type of communication is specified as synchronous (one request/one response, then next request/response) or asynchronous (multiple requests and responses intermixed).

Suppose that the situation is a little more complicated. Let us say that the automobile policies and the homeowner policies are kept in two separate and dissimilar databases. The user request actually needs data from multiple, disparate back-end systems. In this case, there is a need to break the request down into multiple requests (decompose the request) to be sent to the two different back-end databases, then to gather the information that is sent back from the requests, and put this information into the form of a response (recompose). In this case, the Self-Service::Decomposition application pattern (as shown in Figure 3-7) would be more appropriate.

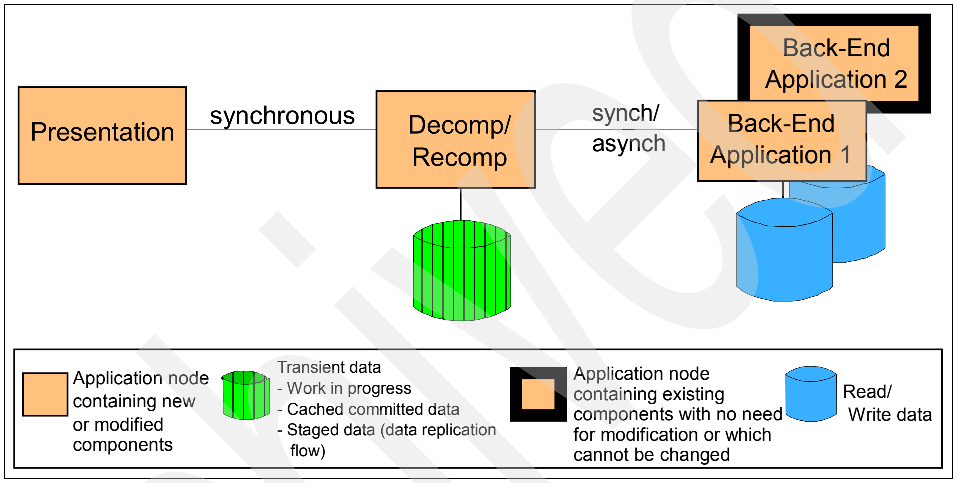

Back-End Application 2
Presentation
synchronous
Decomp/ Recomp
synch/ asynch
Back-End Application 1
Application node containing new or modified components
Transient data
- Work in progress
- Cached committed data
- Staged data (data replication flow)
Application node containing existing components with no need for modification or which cannot be changed
Read/ Write data

Figure 3-7 Self-Service::Decomposition pattern

This Application pattern extends the idea of the application tier that accesses the back-end data by adding decomposition and recomposition capabilities.

3.3.3 Review Runtime patterns

You can refine the Application pattern further with more explicit functions. Each function is associated with a runtime node. In reality, these functions, or nodes, can exist on separate physical machines or can coexist on the same machine. In the Runtime pattern the physical location of the function is not relevant. The focus is on the logical nodes that are required and their placement in the overall network structure.

As an example, let us say that our customer has determined that their solution fits into the Self-Service business pattern and that the Directly Integrated Single Channel pattern is the most descriptive of the situation. The next step is to determine the Runtime pattern that is most appropriate for the situation.

The customer knows that they will have users on the Internet who are accessing their business data; therefore, they require a measure of security. You can implement security at various layers of the application, but the first line of defense is almost always one or more firewalls that define who and what can cross the physical network boundaries into the company network.

The customer also needs to determine the functional nodes that are required to implement the application and security measures. Figure 3-8 shows the Runtime pattern that is one option.

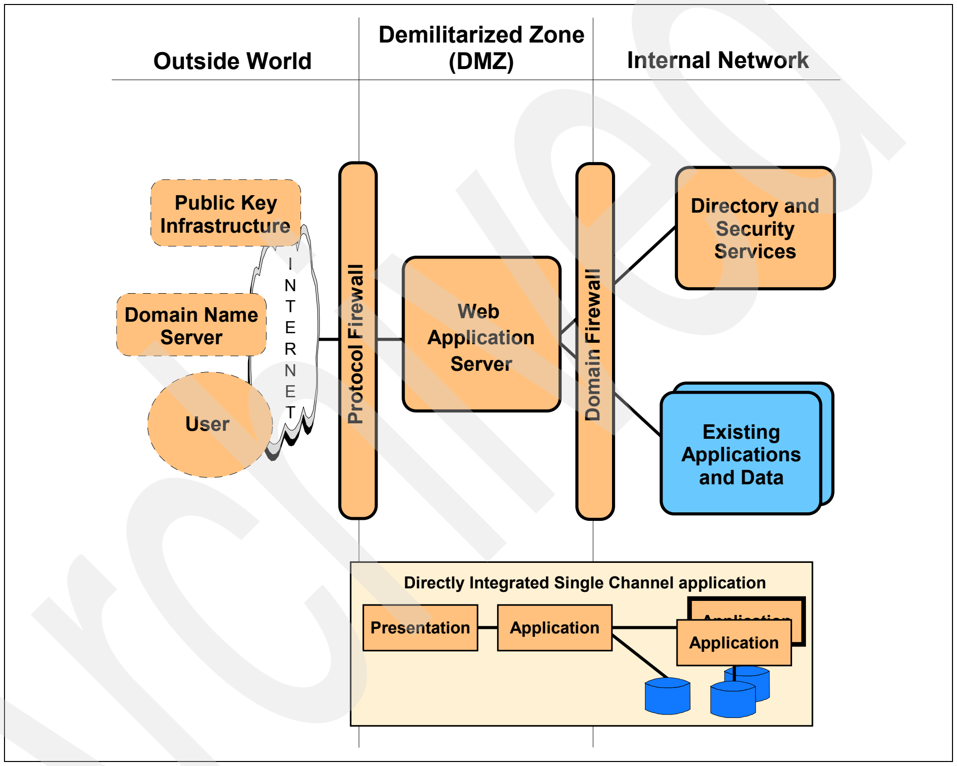


Figure 3-8 Directly Integrated Single Channel application pattern::Runtime pattern

By overlaying the Application pattern on the Runtime pattern, you can see the roles that each functional node fulfills in the application. The presentation and application tiers will be implemented with a Web application server, which combines the functions of an HTTP server and an application server. The Application pattern handles both static and dynamic Web pages.

Application security is handled by the Web application server through the use of a common central directory and security services node.

A characteristic that makes this Runtime pattern different from others is the placement of the Web application server between the two firewalls. Figure 3-9 shows a variation of this pattern. It splits the Web application server into two functional nodes by separating the HTTP server function from the application server. The HTTP server (Web server redirector) provides static Web pages and redirects other requests to the application server. This pattern moves the application server function behind the second firewall, adding further security.

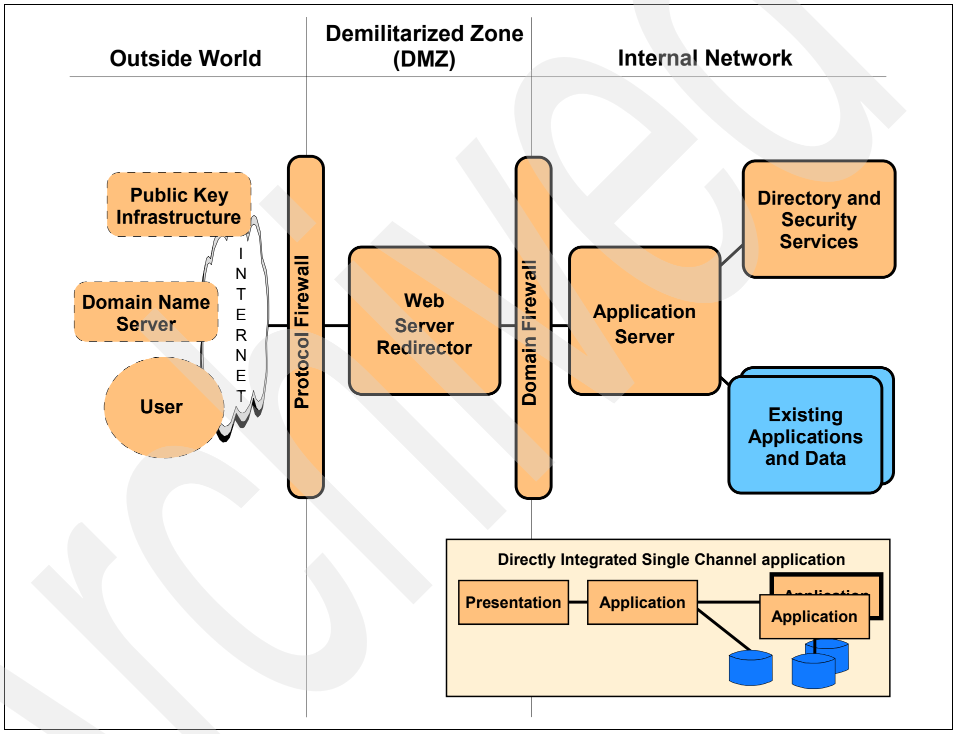


Figure 3-9 Directly Integrated Single Channel application pattern::Runtime pattern

These are just two examples of the possible Runtime patterns that are available. Each Application pattern will have one or more Runtime patterns defined. You can modify these Runtime patterns to suit the customer’s needs. For example, the customer might want to add a load-balancing function and multiple application servers.

3.3.4 Reviewing Product mappings

The last step in defining the network structure for the application is to correlate real products with one or more runtime nodes. The Patterns Web site shows each Runtime pattern with products that have been tested in that capacity. The Product mappings are oriented toward a particular platform. However, it is more likely that the customer will have a variety of platforms involved in the network. In this case, you can mix and match Product mappings (this is dependent on the supported platforms of the IBM products). For example, you could implement the runtime variation in Figure 3-9 on page 78 using the Product mapping depicted in Figure 3-10.

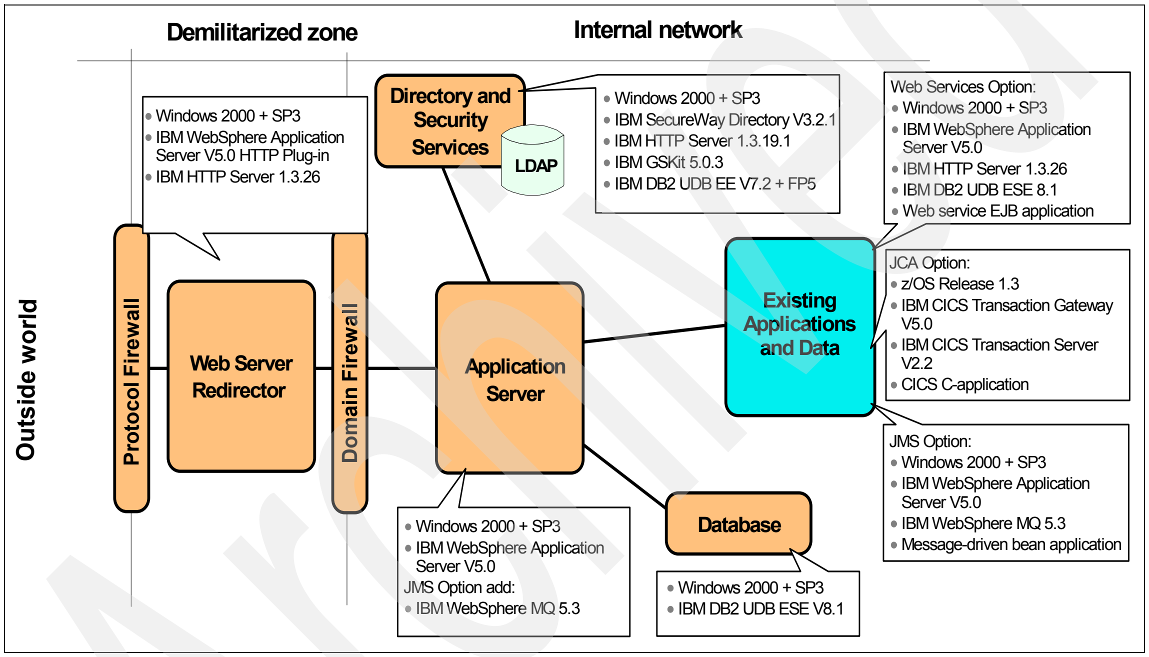


Figure 3-10 Directly Integrated Single Channel application pattern::Product mapping=Windows 2000

3.3.5 Reviewing guidelines and related links

The Application patterns, Runtime patterns, and Product mappings can guide you in defining the application requirements and the network layout. The actual application development has not been addressed yet. The Patterns Web site provides guidelines for each Application pattern, including techniques for developing, implementing, and managing the application, based on the following guidelines:

- Design guidelines provide tips and techniques for designing the applications.
- Development guidelines take you through the process of building the application, from the requirements phase all the way through the testing and rollout phases.
- System management guidelines address the day-to-day operational concerns, including security, backup and recovery, application management, and so forth.
- Performance guidelines give information about how to improve the application and system performance.

3.4 Patterns for e-business naming conventions

The Patterns for e-business use a standard naming convention with the objective of making it easier for the reader to fully identify the referenced asset.

The capitalization convention is to use lower case for pattern and upper case for the first and most significant qualifier, as seen in the following example:

- Business pattern
- Application pattern
- Runtime pattern
- Product mapping

When referencing a specific type of pattern, the higher level qualifier (business) is not capitalized, as seen in the following example:

- Self-Service business pattern

The textual notation Business pattern::Application pattern::Runtime pattern::Product mapping is used to represent the position of an asset within the hierarchy. Occasionally an intermediate level or the pattern type will be omitted for brevity, as seen in the following examples:

- Self-Service::Router application pattern
- Self-Service::Router runtime pattern

In addition, when it is necessary to identify variations or product instances at the same level in the hierarchy, an equals sign (=) will be used, as seen in the following examples:

- Self-Service::Decomposition=Integration Server runtime pattern
- Application Integration::Direct Connection=Message Connection::Product mapping=Web services

3.5 Summary

The IBM Patterns for e-business are a collected set of proven architectures. You can use this repository of assets to facilitate the development of Web-based applications. Patterns for e-business help you understand and analyze complex business problems and break them down into smaller, more manageable functions that you can then implement.

4

Business and IT drivers

Depending on your knowledge level of the Access Integration::Client application patterns, there are different entry points possible for navigating to the appropriate Application pattern or patterns. Some architects may be comfortable with learning the details of each Application pattern found in Chapter 5, “Application patterns” on page 99, and selecting the appropriate pattern based on their knowledge of the Application patterns and customer requirements. In other cases, architects may desire a set of selection criteria used to navigate to the appropriate Application pattern or patterns.

The objective of this chapter is to define the business and IT drivers and capabilities used as criteria for selecting the appropriate Application pattern or patterns.

The chapter is organized into the following sections:

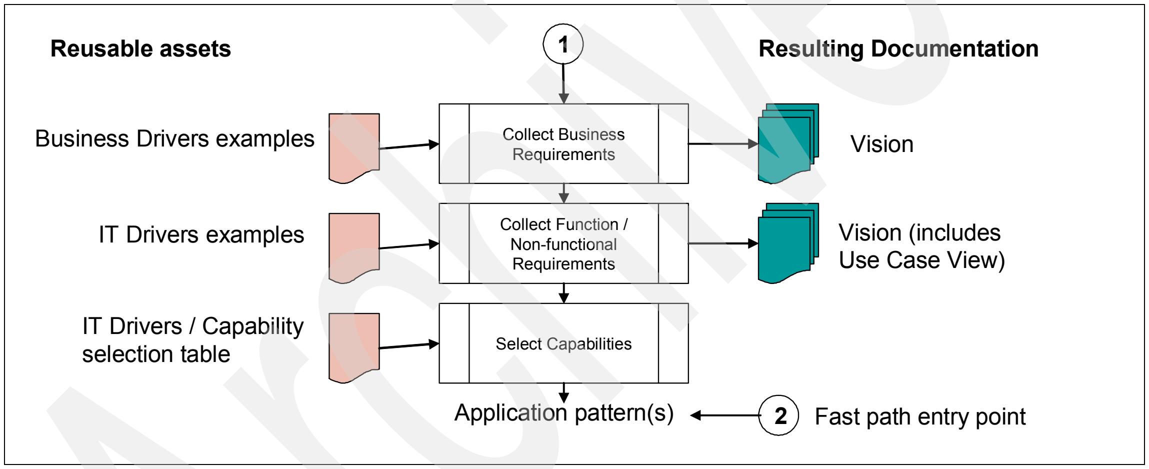
- Overview
- Business drivers
- IT drivers
- Capabilities
- Select capabilities based on IT drivers

4.1 Overview

In many patterns redbooks, the reader learns the details of all of the Application patterns and then from this knowledge selects the proper Application pattern given their customer requirements. This approach is known as the fast entry point (see Fast path entry point 2 in Figure 4-1) and is still available in this book.

In an effort to make the patterns more accessible, we have added IT drivers and capabilities as selection criteria used to select the appropriate Application pattern (see entry point 1 in Figure 4-1).

Note: Figure 4-1 highlights the steps found in this chapter. Refer to 3.1.1, "Process for using patterns within RUP" on page 64, for a description of the entire process.



Reusable assets
1
Resulting Documentation
Business Drivers examples
Collect Business Requirements
Vision
IT Drivers examples
Collect Function / Non-functional Requirements
Vision (includes Use Case View)
IT Drivers / Capability selection table
Select Capabilities
Application pattern(s)
2
Fast path entry point

Figure 4-1 Entry points for selecting the appropriate Application pattern or patterns

We have included a summary description of business drivers, IT drivers, and capabilities to better understand how they are different, and explain the relationship between them.

Business drivers

Business drivers help identify what the core issues are and describe where the business wants to be. In this book the business drivers are included to open a dialog with the customer and provide some common business requirements. We do not include selection criteria from business drivers to Application patterns.

IT drivers

IT drivers are the core functional and non-functional requirements of a customer solution. The IT drivers in this book represent the common requirements captured from customer engagements. The IT drivers can be used as selection criteria to find the appropriate capabilities, which are directly linked to the Application pattern or patterns.

Capabilities

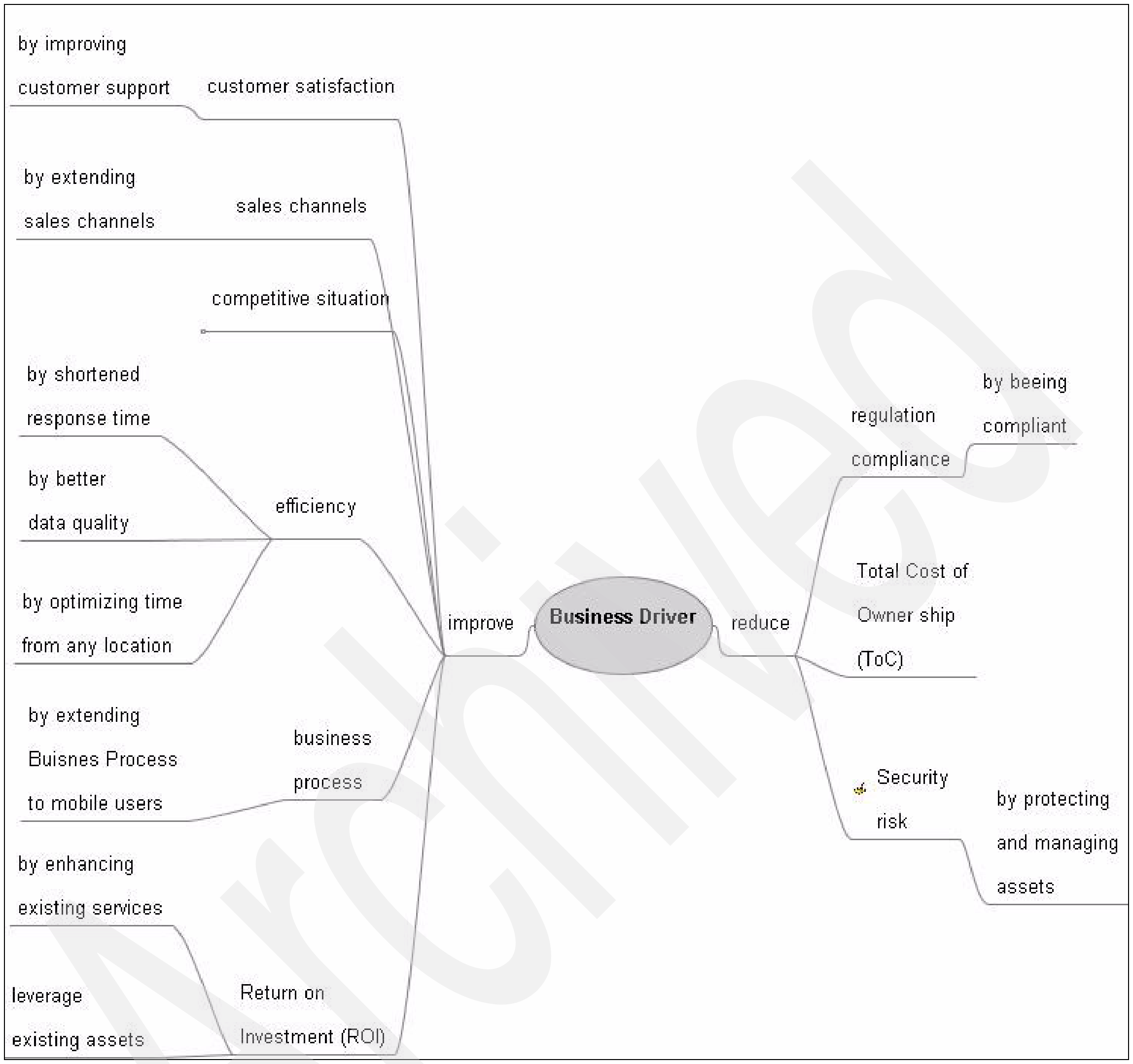
Capabilities are technical helpers to build a bridge from IT drivers to the Application pattern or patterns. They are a collection of the most common services experienced in many mobile projects.

The capabilities map directly to the Application patterns. If you are familiar with the capabilities and can match customer requirements to the capabilities, you can use the fast path entry point (go to Chapter 5, “Application patterns” on page 99).

4.2 Business drivers

Business drivers are used to set an agenda for change and used to gain consensus by assessing the business goals. Business drivers help identify what the core issues are and describe where the business wants to be. Capturing the business drivers is typically a process of analyzing the market and competition, as well as defining desired business capabilities. In a very high condensed form, most business drivers aim to increase revenue or reduce cost.

Figure 4-2 on page 86 depicts a mind-map of common high-level business drivers found within the mobile solution domain.



by improving customer support
customer satisfaction
by extending sales channels
sales channels
competitive situation
by shortened response time
by better data quality
efficiency
by optimizing time from any location
by extending Buisnes Process to mobile users
business process
by enhancing existing services
leverage existing assets
Return on Investment (ROI)
improve
Business Driver
reduce
regulation compliance
by beeing compliant
Total Cost of Owner ship (ToC)
Security risk
by protecting and managing assets

Figure 4-2 Business drivers mind-map

Increase customer satisfaction

Customer satisfaction is a key business driver for customer retention, increased value, and customer loyalty. For example, an auto insurance company currently can only process claims by capturing the information in paper form when on site with the customer. This results in delayed processing of the claim. By enabling the auto claims adjuster to work with the customer on site and capture their claim information electronically and then synchronize the data the enterprise system, the claim can be processed more rapidly. This results in increased customer satisfaction.

Extend and improve revenue streams

Extend business applications to external customers to improve revenue streams. For example, by extending existing applications and services to consumers outside the enterprise provides new and better access to information.

Gain competitive advantage

By providing information in a more timely manner for business decisions, a business can gain a competitive advantage.

Improve efficiency

Improving the data quality and response time for mobile worker improves efficiency. For example, sales people can optimize their travel time by remaining connected to business applications, and sync PIM and e-mail.

Improve business process

Business processes can be improved and optimized to speed up important claiming processing by providing process-critical data to users.

Improve return on investment (ROI)

Improving return on investment can be achieved by leveraging and extending applications to mobile users in support of business operations.

Reduce security and regulation risk

When extending business applications and processes to a wide range of devices and users, security risk must be considered. It is important to understand the constraints of the device as well as the system. This is equally important for individuals as it is for businesses. For example, if your customer contacts and banking information is stored on your device, you need the ability to protect this information in the event of the device being lost or stolen.

4.3 IT drivers

When capturing the IT requirements of a customer, there are two fundamental questions that will drive how the requirements will be analyzed, as well determine how the reusable assets of the patterns are used.

First, does the customer already have an existing device? In many cases, the customer may already have a device and want to leverage this to extend their business operations to mobile users. To a certain degree, the device constraint will greatly define what capabilities are possible. In some cases the constraints of the device can be overcome by adding software to create a device platform that provides for greater capabilities.

Second, if the customer environment has not yet defined the client, then you can proceed with greater flexibility in designing the solution based on the capability needed. This includes the selection of the client device based on the capabilities needed to fulfill the requirements.

This section describes the typical IT drivers representing the common requirements found in mobile solutions. We have grouped the IT requirements as follows:

- Functional requirements

  Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks, or functions the system is required to perform.

- Non-functional requirements

  Non-functional requirements address functionality that influences the underlying systems architecture. These functions are essential for providing a robust and scalable environment (for example, security, device constraints, availability, etc.).

4.3.1 Functional requirements

We have included the common IT drivers that define functional requirements for a couple of reasons:

- Describe common requirements to make IT architects aware of the types of requirements typically requested by customers.
- By defining these functional requirements, we can provide selection criteria for mapping the IT drivers to capabilities and ultimately Application patterns.

Table 4-1 on page 89 includes a summary of the common IT drivers that define the functional requirements. The IT driver number column in Table 4-1 on page 89 is provided for reference lookup purposes. When there is a relationship between the IT driver and other IT drivers defined elsewhere in the table, we include the related IT drivers in parentheses. IT driver IT05, for example, can potentially include IT drivers (1–4).

Note: More detailed descriptions for each of the IT drivers can be found in Appendix B, “Description of IT drivers and capabilities” on page 465.

Table 4-1 Summary of common IT drivers that define functional requirements

IT driver name	Description	IT driver number
Multi device access to Web application	Extending access to corporate Web applications by tailoring the Web content for a class of devices.	IT01
Write once render many	Device-independent authoring to support "write once, render many" to a broad range of users and devices.	IT02
Common programming model	Use of an existing programming model to leverage existing assets, programming skills, and techniques.	IT03
Single sign-on	Provide secure and seamless access to applications and information.	IT04
Role based access, personalized content, customized look	Access to information and applications based on user roles. Ability to customize the look and feel following corporate rules and design guidelines, as well as user preferences.	IT05 + (1–4)
Location Aware Services	Provide location information to applications and users using Location Aware Services.	IT06 + (3)
Voice access	Extend access to information and applications by voice access.	IT07 + (3)
Simple push notification	Provide a simple push notification capability to send simple messages.	IT08 + (3)
Alert user/application of defined state changes	Provide subscribe and publish services to users and applications.	IT09 + (3)
Actionable alerts	Able to send actionable alerts to a device.	IT10
Distribute device configuration	Able to distribute device configurations such as an agent and security settings to devices.	IT11 + (3, 10)
Rollback and recovery	Able to roll back or recover a running task in case of errors.	IT12
Inventory	Collect and maintain inventory information for devices.	IT13
Remote maintenance of middleware	Provide distribution, update, and removal of middleware services on devices.	IT14 + (3, 10, 12, 13)
Remote maintenance of applications	Provide distribution, update, and removal of application services on devices.	IT15 + (3, 10, 12, 13)
Simple offline Web view	Able to cache simple Web data on devices for offline viewing.	IT16 + (IT01, 2, 5)
Offline forms	Able to use forms in disconnected mode.	IT17 + (1, 2, 5, 16)

IT driver name	Description	IT driver number
Disconnectable apps with no GUI	Run a headless (no GUI) application on a device.	IT18 + (3, 11–15)
Disconnectable apps with Web UI	Run disconnectable applications that have a Web User Interface.	IT19 + (3, 11–15, 18)
Disconnectable apps relational database sync	Work with a local database and synchronize occasionally with server.	IT20 + (3)
Disconnectable transactional messaging services	Extend the messaging paradigm by enabling transactional services such as transaction security, message queueing, and assured delivery.	IT21 + (3)
Client Web services	Provide and consume Web services on the device.	IT22 (3)
Standalone rich client UI	Provide a rich client user interface using standard frameworks such as JFace and SWT.	IT23 + (3, 11–15, 20, 22)
Disconnectable rich client UI	Provide applications with a rich client user interface using standard frameworks such as JFace and SWT.	IT24 + (IT23)
Disconnectable applications with multimodal access	Provide a multimodal user interface that supports data and voice.	T25 + (3, 11–15, 20–22)
Provide Instant Messaging	Enable people awareness and instant messaging on devices.	IT26 + (8, 9, 11–14)
Provide offline PIM and e-mail access	Provide synchronization of PIM/EMail on devices.	IT27 + (9, 11–14)
Extend collaboration services	Provide collaboration services on devices.	IT28 + (26, 27)
Optimized remote connection	Provide secure and optimized access to the corporate network.	IT29 + (11)
Support existing devices	Support existing devices.	IT30
Standalone disconnectable relational database	Work with a local database without synchronization.	IT31 + (3)
Disconnectable app w/ performance focus	Disconnectable application with focus on response time and performance.	IT32 + (3)

4.3.2 Non-functional requirements

The non-functional requirements tend to be more customer specific and not as easily provided as selection criteria. We have included the common IT drivers that define non-functional requirements for a couple of reasons:

- To describe common non-functional requirements to make IT architects aware of the types of requirements they will face
- To provide some guidelines on how the non-functional requirements influence the architecture and design
- To highlight the non-functional requirements needed by enterprise customers, that IBM software products fulfill (typically an IBM competitive advantage)

Device constraints

In many cases, the customer may already have a device and want to leverage this to extend their business operations to a wide range of devices (laptop, PDA, mobile phone, embedded). To a certain degree, the device constraint will greatly define what capabilities are possible. Capturing the device and network requirements (in use, or possible) is critical.

Availability

High availability minimizes the risk of an outage, and increases the availability of the system access. A system outage can be costly in terms of lost productivity, lost revenue, and lost customers.

Backup and recovery

It is essential that disaster recovery and backup/restore procedures be properly planned and implemented to meet the high demand for security and recovery.

Capacity estimates and planning

Capacity estimates and deployment planning are important tasks in order to provide good user experience and robust connectivity services. Understanding the application type, network types, throughput, devices, and number of users and actors are key factors.

Performance

Performance planning and measurements of the system access must follow business requirements and service level agreements. Areas to consider are response time, data throughput, and system load, for example.

Quality of service

With ever-increasing bandwidth demands being placed on the network, prioritizing traffic is a growing concern. Quality of service (QoS) offers a practical solution for ensuring the guaranteed delivery of critical data.

Configuration management

Configuration management will gather, organize, and locate configuration information about system access. Configuration management allows you to have up-to-date information about the system configuration.

Environment

Technical, physical, social, and organizational environment.

Extensibility and flexibility

The ability to extend the system access with new services. New emerging technologies and business requirements demand a maximum of flexibility. This goes hand-in-hand with the topic standards below.

Standards

Standards are essential in many aspects of system access. They play a critical role in ensuring safety and quality of the Connection services. Standards enable compatibility of functions and seamless communication. Standards are used in obligatory regulatory compliance requirements for governments and health care in many countries. Examples are:

- International Common Criteria Common Methodology for Information Technology Security Evaluation (ISO 15408)
- Federal Information Processing Standard (FIPS): The US and Canadian government standard for encryption certification

Maintainability

The ease of use to maintain a critical system such as a mobile access service is important. Good maintainability will be a key factor for a robust system.

Scalability

Since business goals and user needs change over time, scalability addresses the ability to react in order to reduce cost and effort.

Security

In order to provide secure access to mobile devices, the system itself must be secure.

Accounting

Collect accounting and billing information for cost calculation.

4.4 Capabilities

Capabilities help to build a bridge from the IT drivers to an Application pattern. They are a collection of the most common services experienced in many mobile projects. The capabilities are the defining technical *capability* that is used as the criteria to select the Application pattern.

Table 4-2 on page 94 includes a summary of the capabilities.

In Table 4-2 on page 94, when a client platform is identified it represents the lowest device functionality that fulfills the capability (device platforms that extend the device also include the capability). For more information about the following client platforms refer to 2.2.2, “Client platforms” on page 38:

- Thin Client (TC)
- Voice Enabled Client (VEC)
- Distributed Presentation Client (DPC)
- Distributed Application Client (DAC)
- Distributed Rich Client (DRC)
- Distributed Collaboration Client (DCC)
- Distributed Multimodal Client (DMC)
- Built-in Client (BIC)

Table 4-2 Summary of capabilities

Capabilities		**Client**
C01	Multi device access - Simple browsing (device independent, dynamic content adaptation)	TC
C02	Multi device access - Role-based browsing (device ind, dyn content adapt, personalized)	TC
C03	Location Aware Service integration (integrate location information)	TC
C04	Voice User Interface (native voice user interface, phone/VoIP)	VEC
C05	Online multi modal access (multimodal access to apps - voice/text)	TC BIC
C06	Receive event notifications synchronously	DCC
C07	Subscription-based event notification (subscribe to event notification, receive notification)	TC
C08	Receive event notifications (receive and process actionable alerts, PIM/e-mail sync)	DAC
C09	Remote management of device configurations (OMADM device - init settings, bootstrap)	DAC
C10	Remote management of middleware	DAC
C11	Remote management of applications (remote install/update/remove applications)	DAC
C12	Disconnectable Web content presentation (offline browsing from cache)	DPC
C13	Disconnectable forms processing (offline forms - submit when connected)	DRC
C14	Disconnectable headless application (disconnectable rich client app without UI, that is, RFID)	DAC
C15	Disconnectable application with local user interface (local Web UI)	DRC
C16	Disconnectable applications using relational sync (local data store, sync with server)	DRC
C17	Disconnectable applications using transactional messaging	DRC
C18	Disconnectable applications using Web services (publish/consume Web services)	DRC
C19	Disconnectable applications using rich user interface (local rich Eclipse UI)	DRC
C20	Disconnectable applications using Multimodal UI (voice/text)	DRC
C21	Instant Messaging and People awareness	DCC
C22	Disconnectable PIM/e-mail access (PIM/e-mail sync)	DCC
C23	Distributed collaboration services (messaging, e-meeting, productivity apps)	DCC
C24	Connectivity services (security, roaming, compression, session mgmt)	All

4.5 Select capabilities based on IT drivers

As our process described in 3.1.1, “Process for using patterns within RUP” on page 64, the architect will collect the IT requirements. The architect can then compare their customer requirements to the common requirements we have documented. Table 4-3 provides selection criteria for mapping IT drivers to capabilities (ultimately mapped to the Application pattern).

The input to this process is to know what IT drivers are needed to meet your requirements. The output is to have a list of capabilities that will be used as the criteria to select the appropriate Application pattern.

Table 4-3 Mapping IT drivers to capabilities

IT driver	**C01 Multi device - Simple browsing**	**C02 Multi device - Role-based browsing**	**C03 Location Aware Service integration**	**C04 Voice User Interface**	**C05 Online multi-modal access**	**C06 Receive event notification**	**C07 Subscribe to event notifications**	**C08 Receive/process actionable alerts**	**C09 Remote mgmt. of configurations**	**C10 Remote mgmt of middleware**	**C11 Remote mgmt of applications**	**C12 Disc. Web content presentation**	**C13 Disc. forms processing**	**C14 Disc. headless application**	**C15 Disc. app. local WEB UI**	**C16 Disc. app. relational sync**	**C17 Disc. app. trxl. messaging**	**C18 Disc. app. Web services**	**C19 Disc. app. local rich UI**	**C20 Disc. app. multimodal UI**	**C21 IM & People awareness**	**C22 Disc. PIM and e-mail access**	**C23 Distributed collaboration**	**C24 Connectivity Services**
IT01 Multi device access to Web application	X																							
IT02 Write once render many	X																							
IT05 Role based access, pzn content, customized look		X																						
IT06 Use/exploit location information			X																					
IT07 Voice access				X																				
IT08 Simple push notification						X																		
IT09 Alert user/application of defined state changes							X									X								

IT driver	C01 Multi device - Simple browsing	C02 Multi device - Role-based browsing	C03 Location Aware Service integration	C04 Voice User Interface	C05 Online multi-modal access	C06 Receive event notification	C07 Subscribe to event notifications	C08 Receive/process actionable alerts	C09 Remote mgmt. of configurations	C10 Remote mgmt of middleware	C11 Remote mgmt of applications	C12 Disc. Web content presentation	C13 Disc. forms processing	C14 Disc. headless application	C15 Disc. app. local WEB UI	C16 Disc. app. relational sync	C17 Disc. app. trxl. messaging	C18 Disc. app. Web services	C19 Disc. app. local rich UI	C20 Disc. app. multimodal UI	C21 IM & People awareness	C22 Disc. PIM and e-mail access	C23 Distributed collaboration	C24 Connectivity Services
IT10 Remotely trigger actions on devices								X																
IT11 Distribute device config									X															
IT12 Transmit error recovery									X	X	X													
IT13 Inventory									X	X	X													
IT14 Remote maintenance of middleware services on device										X														
IT15 Remote maintenance of applications on devices											X													
IT16 Cache Web application for offline viewing												X												
IT17 Provide offline capability for forms based Web apps													X											
IT18 Disconnectable apps with no GUI (headless)														X										
IT19 Disconnectable app with Web UI															X									
IT20 Disconnectable relational database with sync on device																X								
IT21 Disconnectable trans messaging services																	X							
IT22 Client Web services																		X						

IT driver	C01 Multi device - Simple browsing	C02 Multi device - Role-based browsing	C03 Location Aware Service integration	C04 Voice User Interface	C05 Online multi-modal access	C06 Receive event notification	C07 Subscribe to event notifications	C08 Receive/process actionable alerts	C09 Remote mgmt. of configurations	C10 Remote mgmt of middleware	C11 Remote mgmt of applications	C12 Disc. Web content presentation	C13 Disc. forms processing	C14 Disc. headless application	C15 Disc. app. local WEB UI	C16 Disc. app. relational sync	C17 Disc. app. trxl. messaging	C18 Disc. app. Web services	C19 Disc. app. local rich UI	C20 Disc. app. multimodal UI	C21 IM & People awareness	C22 Disc. PIM and e-mail access	C23 Distributed collaboration	C24 Connectivity Services
IT23 Standalone rich client UI using local JFace or SWT																			X					
IT24 Disconnectable rich client UI using local JFace or SWT																			X					
IT25 Disconnectable apps with multimodal access																				X				
IT26 Provide instant messaging																					X			
IT27 Provide offline PIM and e-mail access																						X		
IT28 Extend collaboration services																							X	
IT29 Optimized remote connection																								X
IT32 Disconnectable app with performance focus (response time, scalability)															X	X	X		X					

Table 4-4 on page 98 provides a mapping of device classes (see 2.2.1, “Client device classes” on page 36) to the capabilities. This can be used to better understand the device constraints.

Table 4-4 Mapping device class to capabilities

Device class	C01 Multi device - Simple browsing	C02 Multi device - Role-based browsing	C03 Location Aware Service integration	C04 Voice User Interface	C05 Online multi-modal access	C06 Receive event notification	C07 Subscribe to event notifications	C08 Receive/process actionable alerts	C09 Remote mgmt. of configurations	C10 Remote mgmt of middleware	C11 Remote mgmt of applications	C12 Disc. Web content presentation	C13 Disc. forms processing	C14 Disc. headless application	C15 Disc. app. WEB UI	C16 Disc. app. relational sync	C17 Disc. app. trxl. messaging	C18 Disc. app. Web services	C19 Disc. app. rich UI	C20 Disc. app. multimodal UI	C21 IM & People awareness	C22 Disc. PIM and e-mail access	C23 Distributed collaboration	C24 Connectivity Services
Full Function (FF), that is Laptop, Tablet PC, desktop	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
High Function (HF) that is, High-end PDA (PocketPC + phone and wi-fi).	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Low Function (LF) that is, Mobile phone, low-end PDA	X	X	X	X	X	X	X	X	X	X	X											X		
Special Function (SF) that is, RFID tag, smartcard								X	X	X	X													

Where to go next

Now that you have identified the capabilities needed for your customer requirements, you are ready to select the appropriate Application pattern. Proceed to Chapter 5, “Application patterns” on page 99.

5

Application patterns

The Access Integration patterns include Application patterns, Runtime patterns, and Product mappings centered around extending enterprise applications and business processes to rich client devices (laptops and desktops) as well as low function devices (PDAs, cell phones, RFID readers).

The focus of this chapter is on describing the Access Integration application patterns, as well as providing a decision matrix to select the appropriate Application pattern based on the capabilities needed. Application patterns represent the partitioning of the application logic and data together with the styles of interaction between the logic tiers.

The chapter is organized into the following sections:

- Access Integration pattern overview
- Select the Application pattern
- Access Integration::Client application patterns

5.1 Access Integration pattern overview

The Access Integration pattern describes the recurring application and architecture designs that enable access to one or more Business patterns. In particular, the Access Integration pattern enables access from multiple channels (devices) and integrates the common services required to support a consistent user interface and programming model from rich devices to low function devices. Table 5-1 contains an overview explanation of each of the Application patterns.

As seen in Table 5-1, the Web Single Sign-On, Extended Single Sign-On, and Personalized Delivery Application patterns included in the Access Integration pattern are out of the scope of this book. The remaining subset of Application patterns is referred to as the Access Integration::Client application patterns.

Note: The Pervasive Device Access application pattern found in *Patterns: Pervasive and Rich Device Access Solutions*, SG24-6315, has been replaced by the Application patterns found in this book.

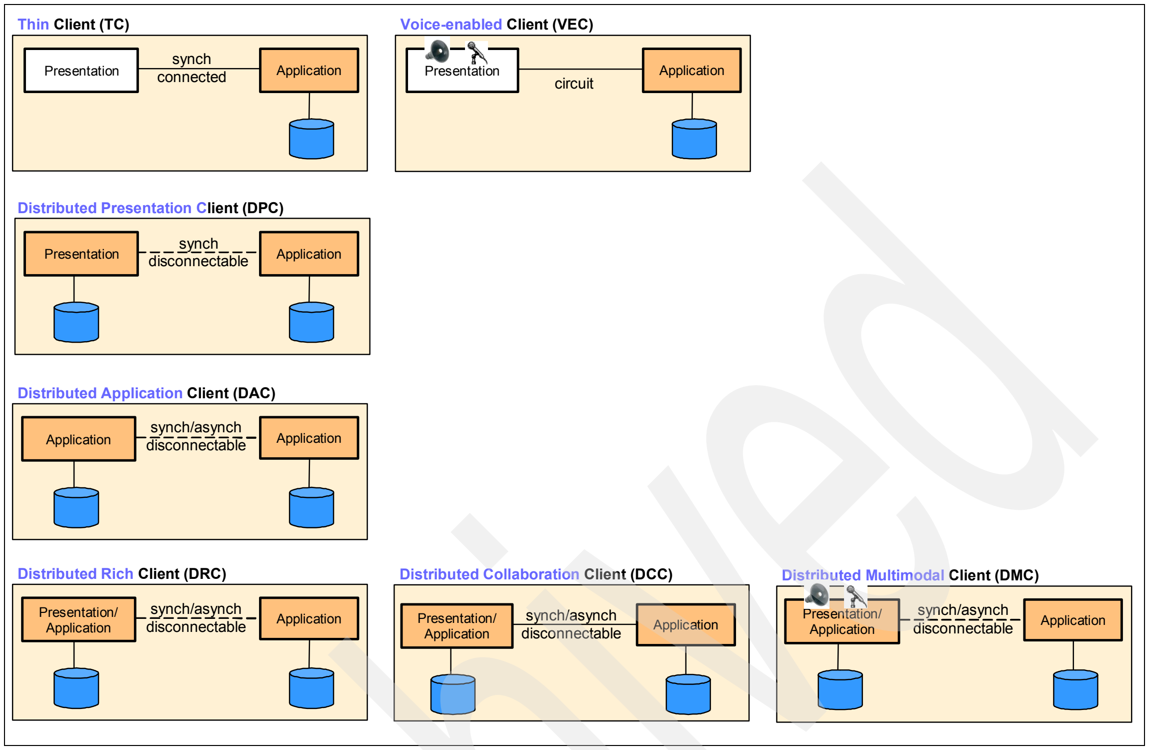
Table 5-1 Summary for the Access Integration::Client application patterns

Application pattern	Description
Web Single Sign-On	These patterns are standard Access Integration application patterns. They are out of the scope of this book and will not be considered further here. For more information refer to the Patterns for e-business Web site: `http://www.ibm.com/developerWorks/patterns`
Extended Single Sign-on	
Personalized Delivery	
Thin Client (TC)	The TC application pattern includes applications that use a browser on the device to access Web-based applications synchronously in an online mode (always connected scenario). The browser is a standard interface and usually provided with the device (for example, HTML, WML browsers).
Voice-Enabled Client (VEC)	In this case, the client is a speech user interface provided by the device (for example, phone). It interacts with a speech-enabled application over circuit-switched (voice capable) networks.
Distributed Presentation Client (DPC)	The DPC represents applications that provide their own presentation layer on the device. This presentation layer is provided by the application and runs on the device. The device accesses the application in online mode, but can also work offline using a local data storage (for example, access Web content online and retrieve content from the local cache in offline mode).

Application pattern	Description
Distributed Application Client (DAC)	The DAC comprises applications that run on the device without providing a user interface (remote *headless application*). Application logic is provided locally so that the application can operate independently from the back-end application logic (operation in connected and disconnected modes). The necessary data storage is also provided locally and is synchronized with the back-end data storage under defined circumstances (time or event triggered).
Distributed Rich Client (DRC)	The DRC is a DAC with a local user interface. This demands higher system resources and implies that a user interacts directly with the application on the device. The user interface ranges from a local Web-based user interfaces to more functional rich user interfaces. The DCC and DMC variations of the DRC Application pattern address specific requirements for collaboration and multimodal applications.
Distributed Collaboration Client (DCC)	The DCC is a variation of the DRC Application pattern. It especially covers online collaboration requirements (for example, instant messaging), as well as disconnectable services (for example, e-mail).
Distributed Multimodal Client (DMC)	The DMC extends the DRC Application pattern with multimodal capabilities (for example, voice navigation in an application).
Built-in Client (BIC)	The BIC is a generic form potentially supporting all of the patterns above. It addresses the fact that certain functions are provided by the device manufacturer through built-in applications. In this case, the application programmer may not need to provide middleware or applications for the device since they are provided by the manufacturer. Depending on the function of the built-in application, this pattern matches one of the other Application patterns (and corresponds to the matching Runtime pattern of the Application pattern).

Figure 5-1 on page 102 and Figure 5-2 on page 102 provide a summary depiction of the Access Integration::Client application patterns relevant to this book. The function set provided by the Application patterns shown in Figure 5-1 on page 102 are incremental. The Thin Client application pattern has the lowest function set. The richest capability is delivered by the Distributed Rich Client application pattern and its variations, Distributed Collaboration Client and Distributed Multimodal Client.

Note: Refer to 2.2, “Client devices classes and platforms” on page 36, for detailed explanations of the different clients.



Thin Client (TC)
Presentation
synch connected
Application
Voice-enabled Client (VEC)
Presentation
circuit
Application
Distributed Presentation Client (DPC)
Presentation
synch disconnectable
Application
Distributed Application Client (DAC)
Application
synch/asynch disconnectable
Application
Distributed Rich Client (DRC)
Presentation/ Application
synch/asynch disconnectable
Application
Distributed Collaboration Client (DCC)
Presentation/ Application
synch/asynch disconnectable
Application
Distributed Multimodal Client (DMC)
Presentation/ Application
synch/asynch disconnectable
Application

Figure 5-1 Summary of Application patterns

The Built-in Client application pattern (see Figure 5-2) is a generic Application pattern that evolves into one of the other Application patterns found in Figure 5-1 depending on the function provided by the built-in application. As industry standard interfaces for basic functions of mobile devices are developed (for example, device management capabilities), it is assumed that more and more functions will be delivered by the device manufacturer as built-in applications. This will reduce the development, operations, and maintenance effort for device access solutions.

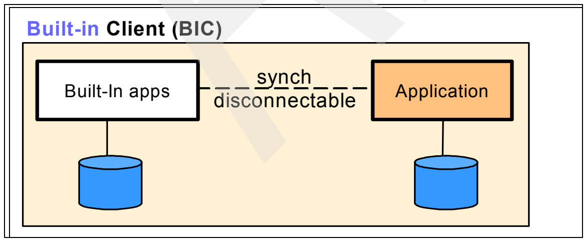


Figure 5-2 Generic Built-In Client application pattern

Detailed information about the Application patterns summarized in Table 5-1 on page 100 can be found in 5.3, “Access Integration::Client application patterns” on page 107. The next section explains how to select the Application pattern or patterns based on the capabilities selection criteria needed to fulfill the customer requirements.

5.2 Select the Application pattern

Figure 5-3 highlights the selection of the Application pattern based on the capabilities selection criteria. The capabilities were previously derived from the IT drivers in 4.5, “Select capabilities based on IT drivers” on page 95.

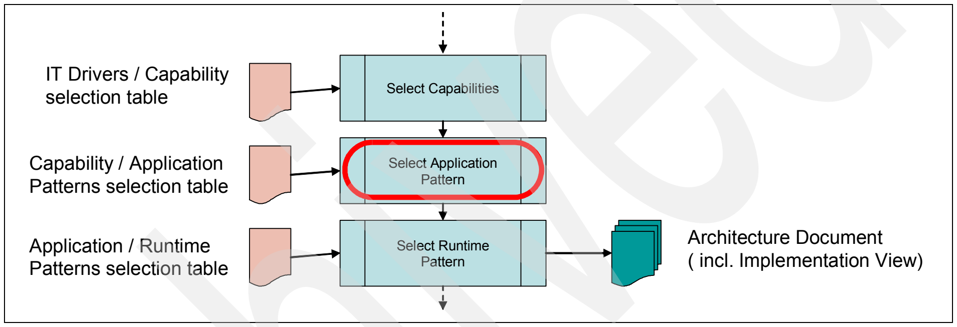

IT Drivers / Capability selection table
Select Capabilities
Capability / Application Patterns selection table
Select Application Pattern
Application / Runtime Patterns selection table
Select Runtime Pattern
Architecture Document (incl. Implementation View)

Figure 5-3 Select Application patterns in tailored process for working example

Using the Application pattern selection table

Table 5-2 on page 105 provides a decision table for selecting the appropriate Application pattern given the capabilities needed to fulfill the customer requirements.

The process for using the Application pattern selection table is as follows:

1. Ensure capabilities needed to fulfill requirements are documented as input for the Application pattern selection process.

   During a previous step, a sub set of the listed capabilities were derived from the IT Drivers (see 4.5, “Select capabilities based on IT drivers” on page 95).

2. Select the Application pattern or patterns for each capability listed in Table 5-2 on page 105.

   The left-most located ’X’ in the table rows points to the Application pattern with a minimum function set providing the capability. Since there is usually more than one capability that must be fulfilled, this selection process could lead to more than one Application pattern. The other, more right placed ’X’ marks in the table mean that this capability is also covered by other (more functionally rich) Application patterns.

The '(X)' marks are included to cover the possibility that the chosen device client might provide built-in functions and capabilities that allow the use of another Application pattern. For example, the mechanism that is used to deliver the capability *Receive simple event notifications* depends on the device's built-in features. A mobile phone is usually capable of sending and receiving SMS notification. On the contrary, a laptop, which is thought of as more function rich, might not be able to do this without additional hardware.

Another example is the Location Aware Service Integration. These services are mainly used by back-end applications for providing information to the user based on their current location. In order to be able to determine the current location, a device with the appropriate capability is necessary. This could be a low function device such as a mobile phone or a rich function device like a laptop. It depends on the technology that is used to locate the device and the user.

After a list of Application patterns covering the capabilities is retrieved, an Application pattern should be found that is able to deliver all of the required capabilities needed. The additional 'X' and '(X)' marks help to identify the smallest common dominator for a solution. The smallest dominator in this context means that the left-most Application pattern in Table 5-2 on page 105 fulfills the selected capabilities. This avoids an oversizing of the solution. Again, if the devices were selected by the customer before, this Application pattern selection process is heavily influenced by the device capabilities as well (see 4.5, "Select capabilities based on IT drivers" on page 95).

The Application pattern identification should lead to one matching Application pattern. In some situations more than one pattern can be chosen. If this is the case, the Runtime patterns that are associated with the selected Application patterns are then combined to create composite variations.

For example, the IT requirements were collected from the customer and mapped to capabilities using Table 4-3 on page 95. The following three capabilities were selected:

- Multi device access - simple browsing (C01)
- Remote management of device configuration (C09)
- Disconnectable Web content presentation - including offline browsing (C12)

Capability C01 leads to the Thin Client application pattern. The Distributed Presentation Client application pattern is the first pattern (found in the table row starting from the left) that provides capability C12 and also C09 (we assume that the chosen device has built-in remote device configuration). This means that we selected two Application patterns:

- Thin Client application pattern
- Distributed Presentation Client application pattern

6.2, "Select Runtime pattern and Product mapping" on page 134, explains how these Application patterns are mapped to Runtime patterns (for this example, Thin Client::Runtime pattern and Distributed Presentation Client::Runtime pattern). Both Runtime patterns are then combined to create a composite Runtime pattern, which provides the required functions.

Table 5-2 Summary of Application patterns that fulfill the defined capabilities

	Capabilities	**Application patterns**						
	X - capability provided by the Application pattern (X) - capability depends on device and connectivity features	**Thin Client**	**Voice-Enabled Client**	**Distributed Presentation Client**	**Distributed Application Client**	**Distributed Rich Client**	**Distributed Collaboration Client**	**Distributed Multimodal Client**
C01	Multi device access - Simple browsing	X						
C02	Multi device access - Role-based browsing	X						
C03	Location Aware Service integration	(X)	(X)	(X)	(X)	(X)	(X)	(X)
C04	Voice User Interface		X					X
C05	Online multi-modal access							X
C06	Receive simple event notifications (for example, SMS)[a]	(X)	(X)	(X)	(X)	(X)	(X)	(X)
C07	Subscription based event notifications	(X)	(X)	(X)	(X)	(X)	(X)	(X)
C08	Receive and process actionable alerts	(X)	(X)	(X)	(X)	(X)	(X)	(X)
C09	Remote management of device configurations	(X)		X	X	X	X	X
C10	Remote management of middleware			X	X	X	X	X
C11	Remote management of applications				X	X	X	X

	Capabilities	**Application patterns**						
	X - capability provided by the Application pattern (X) - capability depends on device and connectivity features	**Thin Client**	**Voice-Enabled Client**	**Distributed Presentation Client**	**Distributed Application Client**	**Distributed Rich Client**	**Distributed Collaboration Client**	**Distributed Multimodal Client**
C12	Disconnectable Web content presentation (incl. offline browsing)			X		X	X	X
C13	Disconnectable forms processing (incl. offline forms)			X		X	X	X
C14	Disconnectable application without UI (headless)				X			
C15	Disconnectable application with WEB UI					X	X	X
C16	Disconnectable application using rich UI					X	X	X
C17	Disconnectable application using multimodal UI							X
C18	Disconnectable application using relational sync				X	X	X	X
C19	Disconnectable application using transactional messaging				X	X	X	X
C20	Disconnectable application using Web services				X	X	X	X
C21	Instant messaging and People awareness						X	
C22	Disconnectable PIM and e-mail access						X	
C23	Distributed collaboration services						X	
C24	Connectivity services	X	X	X	X	X	X	X

a. Capabilities C06–08 heavily depend on the device and the delivery channel of the notification. For example, SMS can be received by Thin Client devices like mobile phones. On the other hand, current laptop models might need special connectivity extensions to be able to receive SMS.

5.3 Access Integration::Client application patterns

This section describes the Access Integration::Client application patterns in more detail. The Application patterns represent the logical components that make up the solution and how these components interact to identify the goal of the application.

For each Application pattern, we provide following information:

- Key IT drivers and capabilities that can be used in deciding the appropriate Application pattern or patterns for your customer requirements.
- Define the solution.
- Provide guidelines for using the Application pattern.
- Describe the benefits and limitations.
- Provide a usage scenario.
- Where to go next.

5.3.1 Thin Client

The Thin Client (TC) application pattern is used to provide consistent access to Web applications from multiple device types (see 2.2.1, "Client device classes" on page 36). This pattern assumes that the device, which is used for accessing a back-end Web application, has a built-in user interface (Presentation component in Figure 5-4 on page 108). This might be a simple WAP or HTML browser on a mobile phone or a rich user interface with browser capabilities such as Eclipse. The back-end application is accessed synchronously in online mode (always connected).

Note: It should be noted that the distributed clients (Eclipse based) include a Web browser and can also be used as Thin Clients.

The Thin Client application pattern assumes that the content provided by the back-end application is prepared (rendered) appropriately for the capabilities of the presentation component on the device (for example, provides WML for WAP browser on mobile phones as well as HTML for standard Web browsers (Firefox, Internet Explorer). This includes the extension of existing Web applications to mobile devices such as mobile phones and PDAs.

The additionally required functions for this content adaptation are addressed by the corresponding Runtime pattern (see 7.2, "Thin Client::Runtime pattern=portal variation" on page 158).

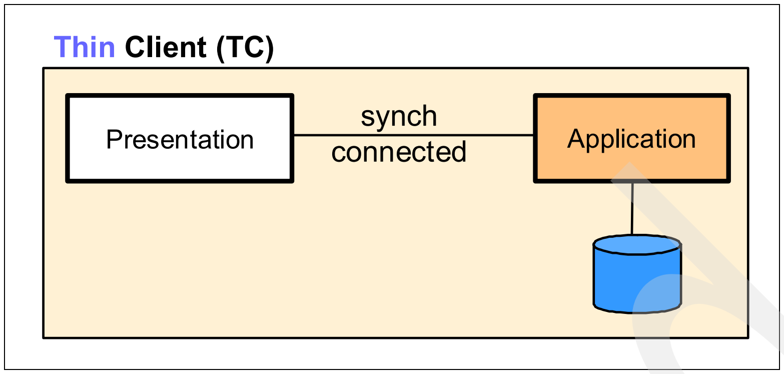


Figure 5-4 Thin Client application pattern

Business and IT drivers

Example business drivers:

- Provide universal access to information and services.
- Time to market.
- Reduce Total Cost of Ownership (TCO).

Example IT drivers:

- Multi device access to Web application
  - Simple browsing
  - Role-based browsing
- Write once render many
- Device capability dependent
  - Distribute device configuration
  - Configuration job transmission error recovery
  - Device inventory

Solution

The Thin Client application pattern consists of three components (see Figure 5-4):

- Presentation component on the client device
- Application component (back-end)
- Data storage (back-end)

The presentation component is the user interface component on a device (such as a WAP browser on a mobile phone or a Web browser on a laptop). It renders data of certain formats depending on its capabilities (for example, WML and iMode format for mobile phone browsers).

Application logic is delivered by the application component located in the back-end. It also manipulates data supplied by the data storage component (in the back-end) and delivers the data to the presentation component on the device. The application component may represent a new application, a modified existing application, or an unmodified existing application. Predominantly these are browser-based applications that must be made available on wireless devices.

Additional services are required to prepare the data stream and to adapt the content to the capabilities of the presentation component (the devices user interface). Security and administration services are necessary to ensure that the device users can achieve a single sign-on to existing applications. These applications may represent applications that automate Self-Service, Collaboration, or Information Aggregation.

These additional services are especially covered by the corresponding Runtime pattern described in 7.2, "Thin Client::Runtime pattern=portal variation" on page 158.

Guidelines for use

The Thin Client application pattern should be used for new or existing Web-based applications that must be made available on mobile devices. Role-based access is covered by the extension of Portal applications to mobile devices. Depending on the number and capabilities of (mobile) devices that should be supported, appropriate products for Thin Client extension services must be chosen.

In general, a larger variety of devices with different capabilities increases the efforts for maintenance and operation. For this reason, we recommend to enterprises that are in control of their device strategy to only support a few device types with similar capability (for example, WML, HTML, and PDA devices with similar screen sizes and browsers). This does not apply to Service Providers, which must support as many device types (and customers) as possible.

Benefits

The Thin Client application pattern gives users a considerable choice of devices to access their applications and data sources (low function to rich function devices; see 2.2.1, "Client device classes" on page 36).

Limitations

The Thin Client application pattern is unlikely to optimize the user interface for any particular device type. If this is required, additional application changes will be necessary.

Scenario

An insurance company has a team of claims assessors visiting policy holders to check the validity and value of their insurance claims. The claims assessors need frequent and fast access to the policy holder policies, claims details, and so on. In addition, they need to initiate contacts with auto repair garages and rental car companies through their extended enterprise applications. While on the road their preferred access is through a wireless-connected Smart phone device. At home or in the office they use a laptop computer for general activities like writing reports. Hence, the insurance company chose the Thin Client application pattern to extend the existing claims applications to be accessed through a Smart phone device.

Where to go next

After the Application pattern has been identified, the matching Runtime pattern and suggested Product mappings can be found in 6.2, "Select Runtime pattern and Product mapping" on page 134.

In this case, the corresponding Runtime pattern for the Thin Client application pattern can be found in 7.2, "Thin Client::Runtime pattern=portal variation" on page 158.

5.3.2 Distributed Presentation Client

The Distributed Presentation Client (DPC) application pattern describes applications that provide their own user interface for the client device. This pattern extends the Thin Client application pattern with its own application-specific user interface (presentation component). This application pattern also allows offline operation (disconnected operation), in addition to synchronous access to the back-end application component in online mode. Local data storage is used on the client to enable continuous operation with a subset of data in offline mode. Additional services assure that the local data subset is in sync with the back-end application data.

The presentation component could also be a standard browser that is extended with a cache and a cache synchronization feature, for example, a Web site made available offline on the device (copied to the local browser cache). The browser cache works as the local data store from where the Web sites can be retrieved when an online connection is not available. Although the cached Web site might not contain the latest available information, it might still be useful information to the user. The Web site cache can be refreshed when an online connection is available again. Simple Web forms could also be made available offline in the cache. A user can fill them in and submit them locally in disconnected mode. When a connection to back-end is available, the locally submitted forms can be sent to the back-end application for submission. Depending on the device's browser capabilities, simple field validation functions might be provided to ensure correct data input (for example, JavaScript capable browser).

The Distributed Presentation Client application pattern also represents applications with user interfaces on the device that use standard functions or even device built-in functions.

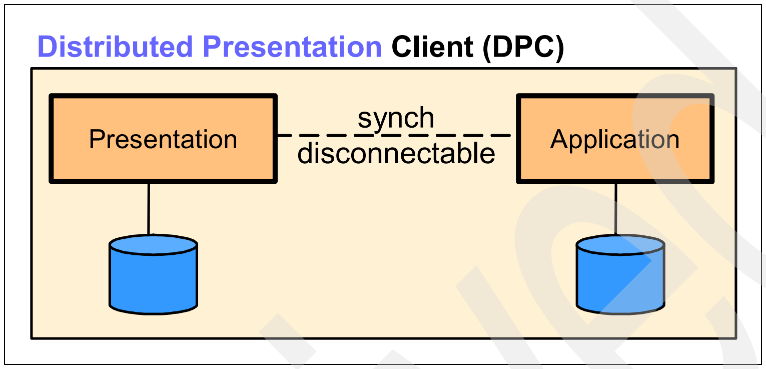


Figure 5-5 Distributed Presentation Client application pattern

Business and IT drivers

The business and IT drivers are as follows:

- Cache Web site content from Web applications for offline viewing.
- Provide offline capabilities for forms-based Web applications.
- Remote device management functions:
  - Recovery in case of configuration job transmission errors
  - Inventory
  - Remote maintenance of middleware

Solution

The Distributed Presentation Client application pattern provides the application programmer with features to make Web content offline available as well as uses existing application logic on the device. The programmer can use a locally available data store, which is kept in sync with the back-end data source through additionally available client services (Distributed client and extension services; see 7.3, “Distributed Presentation Client::Runtime pattern” on page 161). The local user interface on the device must be developed to present data in offline and online mode. And even this user interface could be realized by a browser on the device (for example, for offline browsing and offline forms).

Guidelines for use

The Distributed Presentation Client application pattern should be used for new or existing Web-based applications, which must be made available on mobile devices in connected and disconnected modes. The disconnected use is only recommended if the available offline information must only be up-to-date to a certain extent (information is only updated during synchronization of the data storage). The use of offline forms for data capture in disconnected mode should be chosen when only small amounts of data are captured in static forms and application logic is not necessary during the data capture process (only field validation functions).

Content adaptation and rendering for different devices are also addressed by the Distributed client services (see 7.2, “Thin Client::Runtime pattern=portal variation” on page 158).

Benefits

Content provided by existing Web applications can be viewed on mobile devices in online and offline mode. Simple forms might be used to capture data with mobile devices. The mobile devices only need a browser with cache capabilities and/or the Distributed client and extension services for synchronization of the content. The application programmer only has to provide the presentation component (user interface); the application logic runs on the back-end server. This requires less resources on the device, and lowers application operation and maintenance costs. High function devices and major changes to the existing application are usually not necessary (static Web content; see 2.2, “Client devices classes and platforms” on page 36).

The additional required services are provided by the appropriate Runtime pattern.

Limitations

The Web site caching mechanism only works for static Web sites. Forms-based solutions with offline capabilities are restricted by the device’s browsers capabilities and limitations necessary for offline usage (only simple forms). Comprehensive application logic is not provided in offline mode.

Scenario

A insurance company provides driving directions through a Web-based service. The directions are displayed statically by a Web page (including a map and printed driving instructions). Now the salesperson can download the driving directions for their next customer visit to their PDA or Smart phone. This allows the sales person to take this information with them and to look it up without the need of an online connection (or carrying paper maps).

Where to go next

After the Application pattern has been identified, the matching Runtime pattern and suggested Product mappings can be found in 6.2, “Select Runtime pattern and Product mapping” on page 134.

In this case, the corresponding Runtime pattern for the Voice-Enabled Client application pattern can be found in 7.3, “Distributed Presentation Client::Runtime pattern” on page 161.

5.3.3 Distributed Application Client

The Distributed Application Client (DAC) application pattern varies from the Distributed Presentation Client application pattern by providing an application component on the device and the asynchronous transmission option (see Figure 5-6). It does not have a presentation component on the device (headless, no user interface).

The Distributed Application Client application pattern represents custom-developed application scenarios on the client with offline (not always online, or occasionally connected) capabilities. These types of clients process data in a store and forward mode where the data is stored locally until a connection can be made and the data can be transmitted to the server for onward processing. Application logic is provided locally by the application component and can run without the presence of a connection to the back-end application.

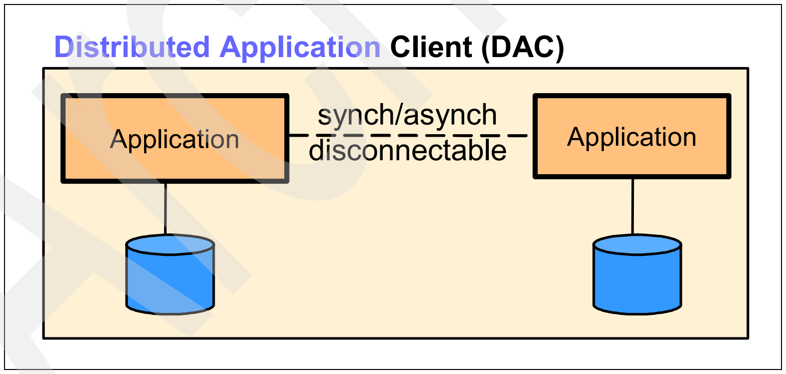


Figure 5-6 Distributed Application Client application pattern

This pattern is usually applied to embedded applications that run on remote devices without local user interfaces (headless). These applications are integrated with back-end applications through a set of services. For example, data can be collected and computed on the embedded device without a back-end connection. Initiated by a trigger, the device connects to the back-end and delivers the locally stored data for further processing by the back-end application. The trigger could be a timer (scheduled synchronization), the reach of a minimum/maximum value (alert), an error/warning, or even a message from the server.

Examples of such embedded devices are RFID readers, set-top boxes, home gateways, routers, in-car management systems, intelligent meter readers, and any kind of remotely controlled machinery.

Since there are many different devices to which this Application pattern applies, it is difficult to find the common denominator and provide a common solution to serve all the different clients. The Application pattern seeks to overcome this problem by targeting a smaller set of devices and defining common client services (see 2.2, “Client devices classes and platforms” on page 36). These services comprise:

- Access services
- Managed client services and Platform Management
- Access to the device hardware

Please refer to 2.2, “Client devices classes and platforms” on page 36, for detailed explanations of the available technologies.

Note: If the application component on the client device is extended with user interface capabilities, this pattern will evolve to the Distributed Rich Client application pattern.

Business and IT drivers

Example drivers include:

- Use/exploit location information.
- Simple push notification.
- Alert application of defined state changes.
- Remotely trigger actions on device.
- Device management:
  - Distribute device configuration.
  - Transmission error recovery.
  - Inventory.
  - Remote maintenance of middleware services on the device.
  - Remote maintenance of applications on device.
- Disconnectable application that does not have a GUI (headless).
- Disconnectable relational database services on device with sync.
- Disconnectable transactional messaging services.
- Disconnectable Web services.

Solution

The Distributed Application Client application pattern focuses on the application component on the client device. The application component is integrated with a back-end application through defined services like the Distributed client and Distributed client extension services (see Distributed Application Client::Runtime pattern in 7.4, "Distributed Application Client::Runtime pattern" on page 164). The client application component uses these services by building on top of a common runtime and common framework that can run on numerous hardware devices and operating systems. The framework and the services ensure seamless back-end integration and offer a common programming model (for details see 2.1, "Programming models" on page 28). For example, the data synchronization or messaging services are provided by the Distributed client and Distributed client extension services and can be easily used by the application on the device.

Benefits

The Distributed Application Client application pattern provides a common programming model for server and client side applications. Distributed Application Client applications can enable clients to operate in both online and offline mode. The applications can also use additional (platform and runtime) services, including transactionality, relational database, reliability, security, and management services (like device management).

Although the extension services provide a stable runtime environment, the application on the device should be designed and developed with great care. A local corrective user interaction on the (usually remotely installed) device is not possible.

Limitations

Caused by the higher resource requirements, the Distributed Application Client applications are limited to a set of devices that are capable of operating the runtime environment together with the framework required by the clients. A local user interface is not available.

The typical instantiation may not contain any local user interface, but a Web container on the system could enable the device to function as an application server, allowing another device to use a browser to access a user interface. For example, a home networking router has no attached screen and is typically not interacted with. However, it does contain an embedded Web application that enables configuration and management via a remote Web browser.

Scenario

A possible scenario is a remote power meter in a household. A local application records the power consumption over time. The amount of consumed power is stored locally and transmitted to the energy provider in a timely fashion. This allows the energy provider to bill the household automatically. It also offers additional use cases, such as identifying peak times, remote power management, and charges for different times of day.

Where to go next

After the Application pattern has been identified, the matching Runtime pattern and suggested Product mappings can be found in 6.2, “Select Runtime pattern and Product mapping” on page 134.

In this case, the corresponding Runtime pattern for the Voice-Enabled Client application pattern can be found in 7.4, “Distributed Application Client::Runtime pattern” on page 164.

5.3.4 Distributed Rich Client

The Distributed Rich Client (DRC) application pattern depicted in Figure 5-7 on page 117 extends the Distributed Application Client application pattern with user interface capabilities. This implies that client devices must fulfill higher resource requirements to provide user interfaces for human interaction (for example, hardware for user interaction such as a display and keyboard, software, and graphical libraries).

The Distributed Rich Client application pattern contains local data storage in order to achieve independence from the back-end application in disconnected mode. Subsets of data are held on the device to be able to continue working when network connectivity is not available or not allowed. Synchronization services ensure that the locally manipulated data is kept in sync with the back-end data storage (for example, through an embedded database synchronization service). The local user interface (presentation component) provides access to the local data storage. The application on the device provides application logic for data manipulation.

The capabilities provided by the application and the presentation components vary in complexity and richness. The user interface can be a local Web user interface running in the local Web container, or a highly sophisticated function rich interface (using Eclipse and extension services). The application component can range from applications with simple field validation functions to applications with complex transactions.

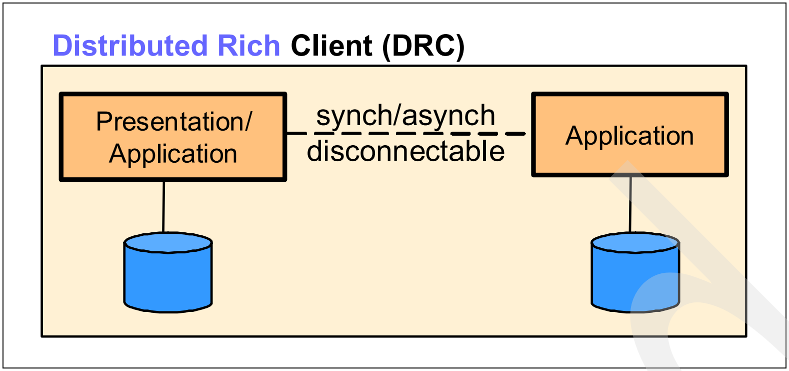


Figure 5-7 Distributed Rich Client application pattern

There are many different (mobile) devices on the market. It can be difficult to determine the common denominator and provide a solution for an intermediary node to serve all the different clients. Devices for the Distributed Rich Client application pattern, therefore, try to overcome this problem by targeting a smaller set of devices and defining common client services (see 2.2, “Client devices classes and platforms” on page 36). These services include:

- Interaction services (graphical user interface)
- Access services
- Managed client services and Platform Management
- Access to the device hardware

The basic Distributed Rich Client application pattern consists of only one node, the client application. This node is capable of providing all the business functionality required.

Overall solution requirements are addressed by the application and Platform Management services (see 2.2, “Client devices classes and platforms” on page 36). For example, the manageability of the rich client application on the device is a mandatory requirement for production environments. Especially for more complex applications, options for remote (maybe unattended) installation, update, configuration, and remote control must be available.

Business and IT drivers

Examples for IT drivers (see Figure 4-3 on page 95):

- Use/exploit location information.
- Simple push notification.
- Alert user/application of defined state changes.
- Remotely trigger actions on device.

- Device management:
  - Distribute device configuration.
  - Transmission error recovery.
  - Inventory.
  - Remote maintenance of middleware services on the device.
  - Remote maintenance of applications on device.
- Disconnectable application that has a Web UI or rich UI.
- Disconnectable relational database services on device with sync.
- Disconnectable transactional messaging services.
- Disconnectable Web services.

Solution

The key to the Distributed Rich Client application pattern is the application itself. As opposed to other Web-based solutions such as the Thin Client application pattern, the DRC focuses on the client tier, which has many of the capabilities of a server platform on the local device to allow for working in a disconnectable mode.

The client application uses services and accesses components on the server side. It is built on top of a common runtime and common framework that can run on numerous hardware devices and operating systems.

Many Distributed Rich Client applications are capable of operating in both online and offline mode. These applications are always available for the user with full functionality, even if the network is down for any reason. Once the network is available, the data and information can be synchronized to/from the server and client. When working in a disconnected mode, it is important to note that only the business function is available all the time, not the most current data.

Guidelines for use

Distributed Rich Clients are similar to the clients of client-server era, in that the devices provide fat client functionality; however, they have the advantage of server-based management. Some of the application logic is implemented on the client-side where they can use services that are not available for thin clients, for example, transactionality, reliability, security, offline data store (relational database), and so on.

Distributed Rich Client applications do not have the usability constraints that Thin Clients have. Interactions can be longer and more complex. Distributed Rich Clients can have controls (user interface components) that are not available to Thin Clients.

Beyond the rich user experience, there are various qualities of service available to these clients. For example, a client and a server can exchange information over a reliable communication, and the user can make sure that the interaction was successful.

Distributed Rich Client applications should ideally be designed in a way so that they can operate in both online and offline networking modes. Imagine the online operation as a special case of the offline operation where information is exchanged (synchronized) instantaneously, not in a long-running store and forward manner.

Benefits

The Distributed Rich Client applications provide a richer and better user experience with extended functionality. Distributed Rich Client applications can enable clients to operate in both online and offline mode. The applications can also use additional services, including transactionality, relational database, reliability, security, and management services (like device management).

Limitations

In light of the higher resource requirements of the Distributed Rich Client, applications are limited to a set of devices that are capable of operating the runtime environment together with the framework required by the clients.

Applications that require constant online connectivity, like instant messaging, or multimodal access are covered by variations of this pattern (see 5.3.5, “Distributed Collaboration Client” on page 120, and 5.3.7, “Distributed Multimodal Client” on page 124).

Scenario

The working example for this book defined in Part 3, “ITSO MobileAdjuster patterns working example” on page 235, features the Distributed Rich Client application pattern.

The scenario is based on the fictitious ITSO insurance company. In this scenario, mobile adjusters can access and change insurance claims from there mobile device while inspecting the car damages at the scene of the accident. This is possible because the function rich Mobile Adjuster application is able to work in a connected or disconnected environment. The underlying technology ensures that necessary data is available in offline mode and synchronized with the back-end regularly. Locally provided application logic guarantees application robustness (for example, correct data input).

Where to go next

After the Application pattern has been identified, the matching Runtime pattern and suggested Product mappings can be found in 6.2, “Select Runtime pattern and Product mapping” on page 134.

In this case, the corresponding Runtime pattern for the Voice-Enabled Client application pattern can be found in 7.5, “Distributed Rich Client::Runtime pattern” on page 169.

5.3.5 Distributed Collaboration Client

The Distributed Collaboration Client (DCC) application pattern is a variation of the Distributed Rich Client application pattern. It especially addresses rich applications that need frequent online connectivity to their server counterpart for collaboration purposes. Examples are instant messaging and Web-based e-mail applications.

The client side consists of one node with presentation and application components. Application logic is locally provided to ensure rich user experience and performance advantages.

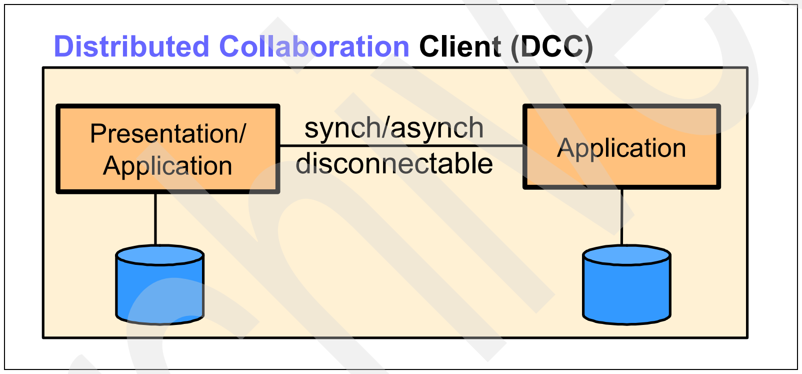

Distributed Collaboration Client (DCC)
Presentation/ Application
synch/asynch
disconnectable
Application

Figure 5-8 Distributed Collaboration Client application pattern

Business and IT drivers

Since this pattern is a variation of the Distributed Collaboration Client application pattern, the same requirements can be applied (see 5.3.4, “Distributed Rich Client” on page 116). In addition, the following IT drivers and capabilities make this pattern unique:

- Instant messaging
- People awareness
- E-mail

Solution

The Distributed Collaboration Client application pattern provides client components for collaboration, giving a rich user experience. The client application offers device-dependent user interface capabilities.

Guidelines for use

The application pattern is used when Distributed Rich Client application pattern IT drivers also include requirements for collaboration functions with online connectivity needed frequently (see 5.3.4, “Distributed Rich Client” on page 116).

Benefits

The Distributed Collaboration Client application patterns provide a richer user experience with extended functionality. Distributed Rich Client application patterns can enable clients to be operated on mobile devices with limited resources. The applications can also use additional services, including transactionality, relational database, reliability, security, and management services (like device management).

Limitations

Caused by the higher resource requirements, the Distributed Collaboration Client applications are limited to a set of devices that are capable of operating the runtime environment together with the framework required by the clients. The major difference and limitation to the Distributed Rich Client is the necessity of a constant online connection for capabilities such as instant messaging and people awareness. The DCC can run in a disconnectable mode for other application common to DRC and e-mail collaboration.

Scenario

An example scenario would be the extension of our working example with instant messaging capabilities. The Mobile Adjuster would get the opportunity to communicate online with the other Adjusters or the back office while being at the accident scene. The back office would be able to inform him instantly of changes or other necessary information.

Where to go next

After the Application pattern has been identified, the matching Runtime pattern and suggested Product mappings can be found in 6.2, “Select Runtime pattern and Product mapping” on page 134.

In this case, the corresponding Runtime pattern for the Voice-Enabled Client application pattern can be found in 7.8, “Distributed Collaboration Client::Runtime pattern” on page 183.

5.3.6 Voice-Enabled Client

The Voice-Enabled Client (VEC) application pattern represents all applications that are accessed through speech user interfaces (see Figure 5-9 on page 122). A telephony client is the most common example that provides a voice user interface (for example, land-line phone, mobile phone, Voice-over-IP telephone client on a computer with speakers and microphone). The voice user interface is used to access a telephony application at the back-end over telephony networks that are circuit switched (for example, PSTN[1], GSM[2]). This means that the client side is assigned a dedicated channel through which it communicates with the back-end application. The back-end application is able to understand voice commands. It converts these commands to computer commands and uses its application logic to process data from the back-end data store. The results are converted back into human understandable speech output signals that are transmitted back to the client user interface.

The back-end application could be an existing Web application or Portal that needs to be accessed through telephony clients. In this case, a voice channel is added to the existing HTML access channel.

Typical examples are telephone information systems that present any kind of event information to a human caller (for example, movie review star rating and show times).

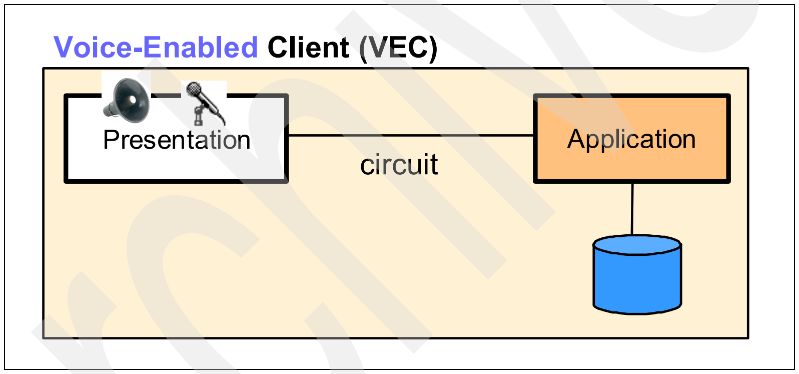


Figure 5-9 Voice-Enabled Client application pattern

Business and IT drivers

Examples for IT drivers:

- Provide voice access to application.
- Add voice access channel to existing application.

Solution

The Voice-Enabled Client application pattern consists of three components:

- Presentation component (speech enabled with microphone and speaker)

[1] PSTN - Public Switched Telephony Network (traditional telephony network)
[2] GSM - Global System for Mobile communication (Cellular network)

- Application component (back-end)
- Data storage (back-end)

The presentation component is a speech-enabled user interface that is built in to the client device. Typically, this is any kind of telephone. Transmission of data is carried out in the form of voice commands and responses over a telephony network.

The back-end application component is able to process voice commands by transforming them into computer commands. It computes the requested results using the back-end data storage. Then the application component translates these results back to the human-understandable voice outputs (audio signals). These are delivered back to the presentation component or user interface where they are presented to the end user.

If an existing back-end (Web) application is extended with a voice access channel, additional components are necessary to provide these capabilities within the application component. The corresponding Voice-enabled client::Runtime pattern provides these through additional components like a Voice Gateway with a Telephony Connector and the Distributed client extension service (see 7.6, “Voice-Enabled Client::Runtime pattern” on page 175). The Voice Gateway with the Telephony Connector provide the functions required to use voice signals for input and output (signal and command translation). The back-end application could still use standard formats like XML for input and output. The Distributed client extension service (voice) takes care of the translation of these data formats to the format of the specific access channel.

Guidelines for use

The Voice-Enabled Client application pattern is used when an application needs to be accessed by telephony type clients. A Web type application that uses standard input and output data like XML is the preferred option. The Distributed client extension service (voice) can use this data and translate it to data for the Voice Gateway. The latter transforms these to voice signals for the presentation component (voice user interface).

It is important to understand that the voice user interface requires special considerations regarding the user interface design. Voice user interfaces must be designed so that users can navigate through menu options without seeing them. If voice recognition is used, the challenges of different languages and dialects must also be considered.

Benefits

The penetration with voice user interfaces (telephony clients) is very high. The number of telephony users is constantly growing thanks to the high availability of mobile/cellular telephony networks.

The user interface client device (telephone) is relatively cheap. It allows human-understandable and hands-free communication (voice), which offers more convenience for certain user groups (for example, drivers). It might even be an option for a user interface where other human-machine interactions are not possible (for example, in certain industries).

Limitations

The voice user interface is more complex than other user interfaces. It requires special design considerations and extensions. It might be expensive if many languages and dialects must be supported.

Scenario

A ticket booking system for a movie theater is accessed through the Internet using a traditional Web browser. At the moment, a central call center is used to allow users to order and reserve tickets over the phone.

Now the ticket company would like to extend the existing browser-based ticket purchasing application with voice capabilities in order to reduce call center costs. For this, the ticket purchasing application, which is a Web-based application producing XML, is extended with voice dialogues in English. The Distributed client extension services (voice) translate this XML content in data formats for the Voice Gateway. Now the end user can directly interact with the Voice Gateway over the phone using the ticket purchasing application rather than needing the call center.

Where to go next

After the Application pattern has been identified, the matching Runtime pattern and suggested Product mappings can be found in 6.2, "Select Runtime pattern and Product mapping" on page 134.

In this case, the corresponding Runtime pattern for the Voice-Enabled Client application pattern can be found in 7.6, "Voice-Enabled Client::Runtime pattern" on page 175.

5.3.7 Distributed Multimodal Client

The Distributed Multimodal Client (DMC) application pattern extends the Distributed Rich Client application pattern with richer user interface capabilities. The use of different input and output channels for application navigation are especially covered by this pattern. For example, a rich client application such as a vehicle navigation system needs to be enhanced with voice navigation features.

The presentation component is extended with additional input and output capabilities (for example, microphone, speaker). The client application component provides application extensions for the additional presentation channels. The application framework might also provide the technology for this richer user interface experience.

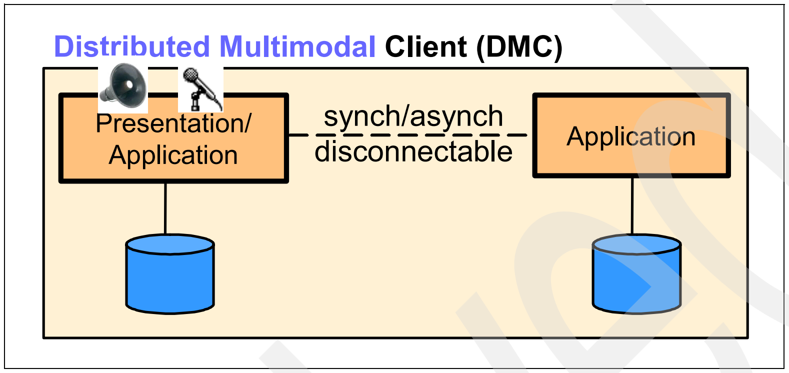

Distributed Multimodal Client (DMC)
Presentation/ Application
synch/asynch
disconnectable
Application

Figure 5-10 Distributed Multimodal Client application pattern

Business and IT drivers

Since the Distributed Multimodal Client application pattern is a variation of the Distributed Rich Client application pattern, the same requirements can be applied (see 5.3.4, “Distributed Rich Client” on page 116). In addition, the following IT drivers and capabilities make this pattern unique:

- Disconnectable application with multimodal user interface

Solution

The Distributed Multimodal Client application pattern provides client components for the extension of rich user interfaces with multimodal capabilities. The combination of voice and text input and output capabilities in one application is provided by additional capabilities in the application and presentation components.

Challenges of the constant connectivity requirement on mobile devices are addressed by management functions of the platform (see 5.3.4, “Distributed Rich Client” on page 116). The client application offers device-dependent user interface capabilities.

Guidelines for use

The Application pattern is used when the Distributed Rich Client application pattern IT drivers also include requirements for multimodal user interface extensions.

Benefits

The Distributed Multimodal Client applications provide an even richer and better user experience with extended input and output functionality. Distributed multimodal client applications can enable clients to be operated on mobile devices with limited resources. The applications can also use additional services, including transactionality, relational database, reliability, security, and management services (like device management).

Limitations

The major difference and limitation of Distributed Multimodal Client is the higher target device requirements. The used device must support the different input and output channels (for example, microphone, speaker, keyboard).

Scenario

An example scenario would be the extension of the ITSO Mobile Adjuster working example with multimodal input and output capabilities. The Mobile Adjuster would be able to communicate with the application voice over and keyboard. While driving to the scene, the adjuster could interact with the application using voice commands to open up and display the customer claim in advance.

Where to go next

After the Application pattern has been identified, the matching Runtime pattern and suggested Product mappings can be found in 6.2, “Select Runtime pattern and Product mapping” on page 134.

In this case, the corresponding Runtime pattern for the Voice-Enabled Client application pattern can be found in 7.7, “Distributed Multimodal Client::Runtime pattern” on page 179.

5.3.8 Built-in Client

The Built-in Client (BIC) application pattern is a generic superset of all other Application patterns introduced in previous sections. The BIC includes functions that are provided by the device manufacturer as built-in applications. Examples are PIM and e-mail clients on PDAs or device management functions on mobile phones (remote configuration). Depending on the provided function of the built-in application, this pattern evolves into one of the other patterns above. The built-in application component will be replaced by the client component of the Application pattern, which matches the IT drivers and capabilities identified for the solution (Table 5-2 on page 105).

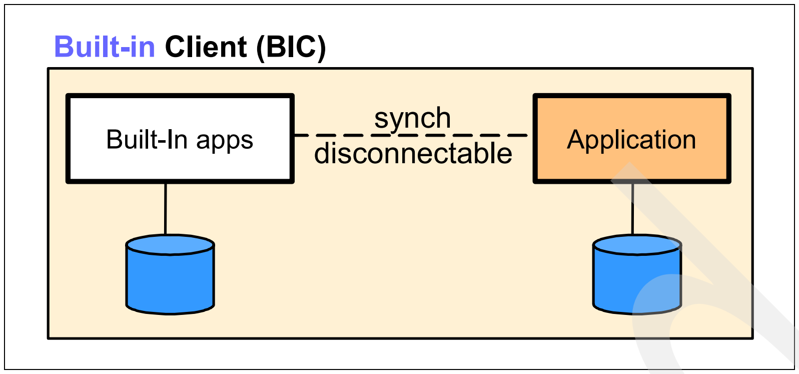


Figure 5-11 Built-in Client application pattern

The fact that the client application might be built in the device does not change the introduced solution development process. The built-in function helps to reduce development costs and potential solution risks.

Business and IT drivers

Examples for IT drivers and capabilities:

- Use devices with existing built-in standard functions and standardized interfaces (for example, device management clients following the OMA-DM[3] standard).

Depending on the built-in function required, the Built-in Client application pattern matches one of the other application patterns, where the appropriate IT drivers and capabilities are listed.

Solution

The Built-in Client application pattern covers all standardized functions that are provided by the device manufacturer as applications on the device. The client-side application does not need to be developed anymore. The rest of the solution depends on the matching Application pattern.

Guidelines for use

This pattern is a generic Application pattern.

Benefits

The Built-in Client is used as an entry point and addresses the advantage that the client-side application has standardized interfaces and is already provided by

[3] OMA-DM = Open Mobile Alliance - Device Management (standard body for device management, http://www.openmobilealliance.org)

the device manufacturer. A client-side application does not need to be developed anymore. The standardized interfaces enable the server-side application to work with the client seamlessly.

Solution development, maintenance, and operation efforts, as well as potential risks, are effectively reduced.

Limitations

In many cases, there is no way to alter built-in function unless at the platform level (software). The other limitations of the matching Application pattern apply.

Scenario

An example is the built-in device management functions of mobile phones. Mobile phones, which include a OMA-DM[4] client, can be configured over the air through the standardized device management interface.

[4] OMA-DM: Open Mobile Alliance - Device Management (standard body for device management, http://www.openmobilealliance.org)

6

Runtime patterns and Product mapping overview

This chapter includes a summary description of the Runtime patterns, Product mapping, and component models. Next we provide criteria to select the appropriate Runtime pattern given the Application pattern.

The chapter is organized into the following sections:

- Runtime patterns and Product mapping summary
- Select Runtime pattern and Product mapping
- Runtime pattern nodes, products, and components

Note: The Runtime pattern, Product mapping, and component model details can be found in Chapter 7, “Runtime patterns and Product mapping details” on page 155.

6.1 Runtime patterns and Product mapping summary

Runtime patterns are blueprints or templates of an architecture that support a chosen Application pattern. They consist of abstract architectural components (nodes) that provide defined encapsulated functionality. The Runtime patterns help to identify these nodes of an architecture that are relevant to fulfill the requirements for a specific customer solution.

Once a suitable Runtime pattern (composite pattern) is identified, the nodes must be instantiated with real products in order to create a real solution. This process was described as the Product mapping in Chapter 3, "Design a solution using the Patterns for e-business" on page 61.

When making the Product mapping selection, we recommend that you consider the following key criteria as well as the non-functional requirements:

- Existing systems and platform investments (including existing devices or device choices)
- Customer and developer skills that are available
- Customer strategy and choice
- Supported platforms for each component
- New extensions that will fit with the existing customer environment

The selected solution should fit into the customer's environment and ensure quality of service, scalability, and reliability so that the solution can grow along with the business and remain in line with the company's strategy (service-oriented architecture strategy, on demand strategy, etc.).

The Product mapping for each of the Runtime patterns is based on IBM software products. In customer engagements, the documented IBM products can be integrated in existing IBM and non-IBM environments. For this reason, we also provide component models of the instantiated Runtime patterns. These help to identify *white spots*, which are areas where the existing customer environment cannot deliver the services recommended in the Runtime pattern. For example, a customer could have an existing OSGi platform deployed on the devices already, but a comprehensive device, application, and Platform Management are not in place yet. The component models show which components from the IBM products provide the non-existent functions.

Since our focus in this book is on SOA Access Integration patterns for devices, special focus is applied to *Distributed client services*, *Distributed client extension services*, and *Thin client extension services* nodes in all the provided Runtime patterns.

Table 6-1 contains a brief description for each of the Runtime patterns and a possible Product mapping. We followed the naming convention of Application pattern::Runtime pattern=(variation) throughout. For information about the patterns notation refer to 3.4, “Patterns for e-business naming conventions” on page 80.

Note: The Runtime pattern, Product mapping, and component model details can be found in Chapter 7, “Runtime patterns and Product mapping details” on page 155.

Table 6-1 Runtime patterns summary for the Access Integration::Client application patterns

Runtime pattern	Runtime pattern description	Product mapping key IBM software
Thin Client::Runtime pattern	This Runtime pattern allows users to access Web information online through their devices (with any type of desktop/laptop). The information is provided through an Application server and delivered by a Web server. **Note:** Runtime environment for standard *Web browser clients*.	► IBM HTTP Server ► IBM WebSphere Application Server (WAS)
Thin Client::Runtime pattern=portal variation	This Pattern is used if in addition to the above role-based and/or personalized aggregated content must be accessed with any (mobile) device. A Portal server with client extensions in the Application server environment provides these functions. **Note:** *Portal-based* runtime environment for *any (mobile) browser* client.	► IBM WebSphere Portal ► IBM WebSphere Everyplace Mobile Portal (WEMP)
Voice-Enabled Client::Runtime pattern	This pattern provides existing applications and data with voice access capabilities. The Voice gateway, Telephony connector nodes and extensions to the Application server node provide these voice interaction capabilities to applications. **Note:** Runtime for *telephony clients*.	► IBM WebSphere Voice Response and Voice Server ► IBM WebSphere Voice Application Access and VoiceXML browser

Runtime pattern	Runtime pattern description	Product mapping key IBM software
Distributed Presentation Client::Runtime pattern	This pattern extends the Thin Client::Runtime patterns with capabilities needed for synchronous disconnectable operation of the client (offline). The Distributed client services and extension services provide services that allow disconnected operation with local copies of data (for example, off-line Web pages, off-line forms). **Note:** Runtime for *any (mobile) browser* client with *offline capabilities*.	▶ IBM WebSphere Everyplace Access (WEA)
Distributed Application Client::Runtime pattern (Headless application)	This pattern provides those Distributed client extension services and Client services that are required to support disconnectable applications using synchronous and asynchronous communication without a local user interface (headless). OSGi as the standardized Java application platform with remote management of applications and services is suggested for the device. Additional services extend the known J2EE programming model to the client side (including HTTP server, Servlet container, messaging services, relational database services, Web services, etc.). The typical instantiation may not contain any local user interface, but a Web container on the system could enable the device to function as an application server, allowing another device to use a browser to access a user interface. For example, a home network router has no attached screen and is typically not interacted with. However, it does contain an embedded Web application that enables configuration and management via a Web browser. **Note:** Runtime for *OSGi based clients without local UI.*	▶ IBM WebSphere Everyplace Deployment (WED) ▶ IBM Workplace Client Technology, Micro Edition (WCTME)

Runtime pattern	Runtime pattern description	Product mapping key IBM software
Distributed Rich Client::Runtime pattern	The Distributed Rich Client extends the Distributed Application Client with the addition of a local user interface (Rich or Web). The Distributed Rich Client runtime pattern provides Distributed client extension services and Client services, which are required to support disconnectable applications using synchronous and asynchronous communication to back-end servers. There are two client device classes for this Runtime pattern: ► Full function devices (that is, laptop): The combination of the full function device and Distributed client extension services allows for a Rich or Web user interface on the local device. The Eclipse standard, which also uses OSGi, is proposed as the Java application platform on the device in order to provide a J2EE-like programming model. ► High function devices (that is, PDA): The high function devices are more limited than full function devices. The Distributed client extension services in this case support a Web-based UI and local UI using SWT and AWT directly without the aggregation capabilities available in Eclipse. **Note:** Runtime for *OSGi* and/or *Eclipse* clients with local Web UI or Rich Eclipse based UI.	Full function Windows/Linux devices: ► IBM WebSphere Everyplace Deployment (client and server) ► IBM Workplace Client Technology, Enterprise Offering (WCTEO) High function devices: ► IBM Workplace Client Technology, Micro Edition (WCTME)
Distributed Multimodal Client::Runtime pattern	This Runtime pattern adds support for the Distributed Multimodal Client application pattern. The Distributed client extension services and Client services extend the application runtime environment with VoiceXML support. This pattern can be used in composition with all other Distributed rich client and collaboration client patterns. **Note:** Runtime for *OSGi* and/or *Eclipse based clients with Multimodal* browser.	► IBM WebSphere Everyplace Deployment (client and server) ► IBM Workplace Client Technology, Micro Edition with Multimodal toolkit

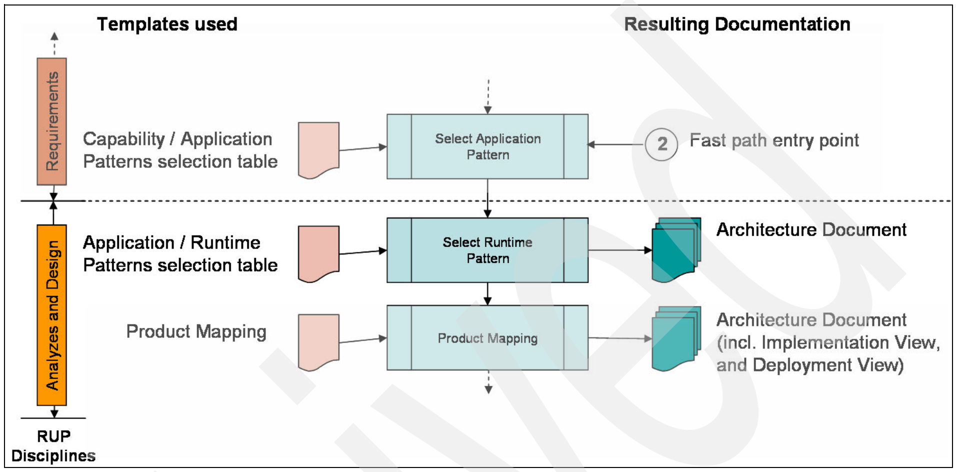
Runtime pattern	Runtime pattern description	Product mapping key IBM software
Distributed Collaboration Client::Runtime pattern	This pattern extends the Distributed Rich Client::Runtime pattern by providing nodes for applications using collaboration functions. Distributed client extension services in conjunction with Distributed collaboration services and Distributed client services deliver these collaboration services for online and/or disconnected operation. **Note:** Disconnected work is not possible for all collaboration scenarios (for example, not for instant messaging). **Hint**: Runtime for *Eclipse-based clients with collaboration* functions.	► IBM Workplace Collaboration Services and Workplace Managed Client™ ► IBM WebSphere Everyplace Deployment ► IBM Workplace Client Technology, Enterprise Offering
Roaming Connectivity:: Runtime pattern	This pattern provides the nodes required to securely access the (server-side) application and runtime environment in the intranet with devices using insecure (public) networks. The Connectivity and Access for Mobile services deliver VPN functions (security), session management, and packet transport optimization. **Note:** Runtime for *clients with VPN* requirements.	► IBM WebSphere Everyplace Connection Manager (server and client)
Composite runtime pattern	This pattern composes other Runtime patterns in case the user requirements cannot be fulfilled by a single pattern. An example would be the extension of the Distributed Rich Client::Runtime pattern with collaboration functions (composition with the Distributed collaboration client) and/or with remote access functions (composition with the Roaming connectivity runtime pattern). **Note:** Runtime for **`Eclipse-based clients with additional functions`** from other runtimes.	(See Product mapping of the combined Runtime patterns.)

6.2 Select Runtime pattern and Product mapping

In 3.1.1, “Process for using patterns within RUP” on page 64, we introduced a tailored process for designing an IT solution using the Patterns for e-business. This section provides a summary of selecting the Runtime pattern and Product mapping, given the chosen Application pattern.

6.2.1 Select the Runtime pattern

The next step after the Application pattern selection, which is explained in Chapter 5, “Application patterns” on page 99, is the *Select Runtime pattern* step highlighted Figure 6-1.



Templates used
Resulting Documentation
Requirements
Capability / Application Patterns selection table
Select Application Pattern
2
Fast path entry point
Analyzes and Design
Application / Runtime Patterns selection table
Select Runtime Pattern
Architecture Document
Product Mapping
Product Mapping
Architecture Document (incl. Implementation View, and Deployment View)
RUP Disciplines

Figure 6-1 Select Runtime Pattern in tailored process for working example

Table 6-2 on page 136 provides a summary of Application patterns (derived from selected capabilities) and the corresponding Runtime patterns.

Each previously chosen Application pattern has a corresponding Runtime pattern. In some cases, there are variations of the Runtime pattern to account for unique runtime characteristics. An Application pattern can be matched to one of the variations or the base Runtime patterns.

Since the richness of the features provided increases from the Runtime pattern in the left column to the one in the right column (see Table 6-2 on page 136), the decision about the finally chosen Runtime pattern is also influenced by non-functional requirements. Examples are the device capabilities or the available budget. A Runtime pattern might be required to fit a certain device with specific limitations that do not allow running rich clients with enhanced UI functions. Or, the available IT budget might bind the solution to a certain device with fewer features and/or limit the set of the instantiated extension services by choosing certain products with less capabilities.

The Composite::Runtime pattern or patterns are combinations of basic Runtime patterns. They are introduced to cover additional requirements that cannot be fulfilled by a single pattern. For example, the Roaming connectivity::runtime pattern can be used together with all other Runtime patterns, because it especially provides the secure access to the server infrastructure.

Table 6-2 Mapping of Application patterns to Runtime patterns

Application pattern	Runtime patterns (richness increases from left to right)							
	Thin Client	Voice-Enabled Client	Distributed Presentation Client	Distributed Application Client	Distributed Rich Client	Distributed Multimodal Client	Distributed Collaboration Client	Roaming Connectivity (additive only)
Thin Client	X		X		X	X	X	X
Voice Enabled Client		X				X		X
Distributed Presentation Client			X					X
Distributed Application Client				X	X	X		X
Distributed Rich Client					X	X		X
Distributed Multimodal Client						X		X
Distributed Collaboration Client							X	X
(Built-in Client)	(X)	(X)	(X)	(X)	(X)	(X)	(X)	(X)

Note: We have included (X) in the Built-in Client row to show that it is possible, but dependent on what is packaged by the device manufacturer.

6.2.2 Select the Product mapping

The next step after the Runtime pattern selection is the *Product mapping* step (highlighted Figure 6-2 on page 137). In Chapter 7, “Runtime patterns and Product mapping details” on page 155, the Runtime patterns are instantiated with IBM software products (that is, Product mapping). In addition, we have included reusable component models for the IBM software products for the most common solution environments. The component models help to identify those solution components on a granular level, which are required to meet the initial solution requirements.

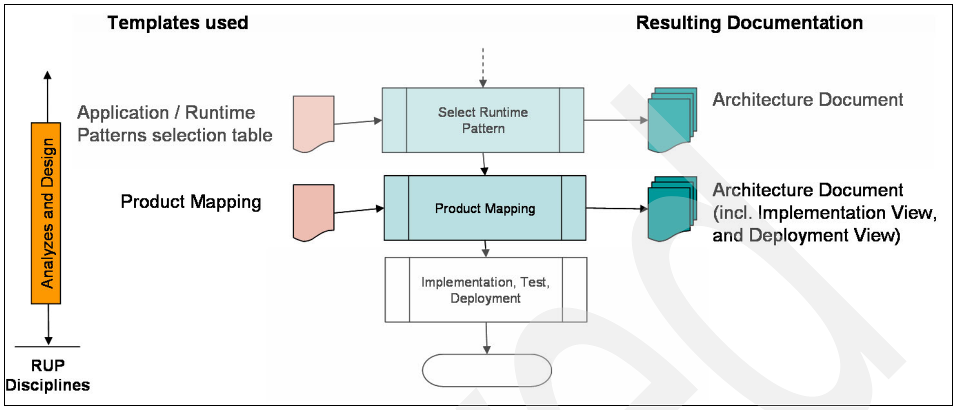

Templates used
Resulting Documentation
Analyzes and Design
RUP Disciplines
Application / Runtime Patterns selection table
Product Mapping
Select Runtime Pattern
Product Mapping
Implementation, Test, Deployment
Architecture Document
Architecture Document (incl. Implementation View, and Deployment View)

Figure 6-2 Product mapping in tailored process for working example

6.3 Runtime pattern nodes, products, and components

This section provides a description of the Runtime pattern nodes, as well as the IBM software products and components used to instantiate the nodes in the Product mapping.

6.3.1 Runtime pattern node descriptions

A Runtime pattern consists of several nodes that represent specific functions. Most Runtime patterns consist of a core set of common nodes with the addition of one or more nodes unique to that pattern.

To better understand the Runtime patterns, we have included a definition for each of the following nodes:

- User node
- Client node
- Distributed client services node
- ISP Gateway (Mobile services) node
- Protocol firewall node
- Connectivity and access for Mobile services node
- Web server redirector node
- Telephony connector
- Voice gateway node
- Presentation server node
- Personalization server node

- Directory and security services node
- Application server node
- Distributed client extension services node
- Thin client extension services node
- Existing data and applications node
- Database node
- Collaboration server node
- Distributed collaboration services node

User node

The user node is most commonly a person who wants to use a specific device for accessing a business application on a back-end server from his local device in either connected or disconnected mode (on/offline). The user node could also be another client or a form of automated component such as an RFID reader.

Client node

The client node could be any device that provides network connectivity such as LAN, WAN, mobile networks (GSM/GPRS/CDMA, etc.), or telephone networks (h.323, etc.). Clients usually contain a CPU, storage to process and to interact with as well as store/request data. Clients might also have a user interface (UI). Most frequently, the client node is a personal computer (desktop or laptop), an embedded headless device (like a RFID reader), Personal Digital Assistant (PDA), smartphone, or another handheld device.

Distributed client services node

Note: This node was formerly called the *Pervasive client services node* in previous redbooks.

This node contains those components that are designed to support device applications and interactions. The distributed client services node is linked very tightly with the distributed client extension services. It contains those components on the device (client) that interact with the server side. This node also includes the layer that is responsible for running the distributed applications on the device. It might provide a complete application platform including a rich user interface.

ISP Gateway (Mobile services) node

The Internet Service Provider (ISP) Gateway provides a network access point to certain types of devices. An example would be using a GPRS connection on a Blackberry device. ISP Gateways can contain the device recognition and user authentication information or they could act as a proxy to provide a communication channel between the device and the server.

Protocol firewall node

A firewall is a hardware and software system that manages the flow of information between the Internet and an organization's private network. Firewalls can prevent unauthorized Internet users from accessing private networks that are connected to the Internet and can block some virus attacks. A firewall can also separate two or more logical parts of a local network to control data exchange between departments. Components of firewalls include filters or screens, each of which controls the transmission of certain classes of traffic. Firewalls provide the first line of defense for protecting private information, but comprehensive security systems combine firewalls with encryption and other complementary services, such as content filtering and intrusion detection.

Firewalls control access from a less trusted network to a more trusted network. Traditional implementations of firewall services include:

- Screening routers (the protocol firewall)
- Application gateways (the domain firewall)

A pair of firewall nodes provides increasing levels of protection at the expense of increasing computing resource requirements. The protocol firewall is typically implemented as an IP router.

Connectivity and access for Mobile services node

This node allows an encrypted, secured, and trusted channel between client devices and corporate back-ends. It also offers seamless roaming to the corporate intranet from any location. In certain cases, this node can be used for decreasing data transmission over the wireless networks through various compression techniques. This is not a mandatory service, but security and connectivity requirements may make it necessary.

Web server redirector node

In order to separate the Web server from the application server, a so-called Web server redirector node (or just redirector for short) is introduced. The Web server redirector is used in conjunction with a Web server. The Web Server serves HTTP pages, and the redirector forwards servlet and JSP requests to the application servers. The advantage of using a redirector is that you can move the application sever behind the domain firewall into the secure network, where it is more protected than within the DMZ.

Telephony connector

The telephony connector node is used to maintain voice-related information such as vocabularies of words, pronunciations of words, and statistics on how certain words are used. It also contains the logic for speech recognition and can convert audio into text.

This node is required to connect the telephony network to enterprise telephony. The node would include any switch (PBX) and might support analogue, digital (E1/T1), and VoIP connectivity depending on enterprise requirements.

Voice gateway node

The voice gateway node supports the call control and management via an IVR and speech technology set (speech recognition and TTS). In our current partnerships we support Genesys Enterprise Edition, Avaya IR, and a range of Cisco products. We also support WebSphere Voice Response V3.1. There are other partners who are also providing IVR connectivity to the IBM speech resources. It is valid to say that any VoiceXML 2.0 and MRCP client compatible system can be supported on this node.

Presentation server node

The presentation server node provides services to enable a unified user interface. It is responsible for all presentation-related activity independent of local client-based applications. In its simplest form, it serves HTML pages and runs servlets and JSPs. For more advanced patterns, it acts as a portal and provides the access integration services (single sign-on, for example). It interacts with the personalization server node to customize the presentation based on the individual user preferences or based on the user's role.

Personalization server node

The personalization server node works with the Web presentation server node to customize the presentation with data that matches the users interests. The personalization server identifies the type or class of the user based on information available about the user. Based on this classification, data taken from a content datastore either in the Personalization tier or from other back-end sources is selected for presentation to the user. It provides the mapping function of user classification to content data.

Directory and security services node

The directory and security services node supplies information about the location, capabilities, and attributes (including user ID/password pairs and certificates) of resources and users known to the Web-based application system. This node can supply information for various security services (authentication and authorization) and can also perform the actual security processing, for example, to verify certificates. The authentication in most current implementations validates the access to the Web application server part of the Web server, but this node also authenticates for access to the database server and synchronization services.

Application server node

The application server node provides the infrastructure for application logic and can be part of a Web application server. It is capable of running both presentation and business logic, but generally does not serve HTTP requests. When used with a Web server redirector, the application server node can run both presentation and business logic. In other situations, it can be used for business logic only. This application server node is the base for the pervasive extension services node.

Distributed client extension services node

Note: This node was formerly called the *Pervasive extension services node* in previous redbooks.

This node extends the device access capabilities of other existing nodes on the server side in the architecture. It is especially designed to interact with the client services on (pervasive) devices by providing the client-side applications and services with transparent access to their peers on the server side. It takes care of data transfer, content adaptation, rendering (cHTML, PDA, VoiceXML), synchronization, device management, etc. Components for notification are also included in this node. This node is invoked based on the commands that the Distributed client extension services node sends, and returns the needed responses back to the client. Examples of this include data updates, query results, instant notifications, and confirmations.

Thin client extension services node

Note: This node was a part of the *Pervasive extension services node* in previous redbooks.

This node is an extension to the Portal server. It especially addresses rendering and content adaptation for many different devices and browser types through device-independent authoring (write once, render many). It allows new devices to be introduced without changes to the style guide or impact to the existing applications or products. To achieve this, the node intercepts the requests issued by the browser and identifies the user agent. This information is used to retrieve the rendering information (policies) about the specific device and browser from the device repository database. Then the response for the device is assembled based on the information found in the device repository.

Existing data and applications node

Existing applications are run and maintained on nodes that are installed on the internal network. These applications provide for business logic that uses data maintained in the internal network. The number and topology of these existing application and data nodes is dependent on the particular configuration used by these legacy systems.

Database node

The function of this node is to provide persistent data storage and retrieval in support of the user-to-business transactional interaction. The data stored is relevant to the specific business interaction, for example, bank balance, insurance information, and current purchases by the user.

It is important to note that the mode of database access is perhaps the most important factor determining the performance of this Web application in all but the simplest cases. The recommended approach is to collapse the database accesses into a single or a few calls. One approach for achieving this is by coding and invoking stored procedure calls on the database.

Collaboration server node

This node represents the existing e-mail and PIM server. In most cases this would be a Lotus® Domino® or Microsoft Exchange server. In a service-oriented architecture environment, this node talks to the Application server node and the Distributed client extension services node. In some scenarios it is also possible to connect to this node without any other nodes. An example of this is that it is possible to configure many smart phones to use the IMAP protocol, which can directly connect into the collaboration server node to retrieve e-mail messages if it supports IMAP connections.

Distributed collaboration services node

This node delivers collaboration extensions to the Portal server environment. In doing this, the Distributed collaboration services node extents an existing Portal-based architecture with collaboration functions. It replaces traditional stand-alone collaboration servers and integrates their functions into a J2EE-based service-oriented architecture. For example, Team collaboration services (awareness, instant messaging, etc.), Document services, and Messaging services (PIM/e-mail) are provided by this node. Special device access capabilities might be addressed by functions in this node or by using it in conjunction with the Distributed or Thin client extension services node.

6.3.2 IBM software products used in Product mappings

All provided Runtime patterns can be instantiated with IBM software products. Although a solution based on the IBM set of products would provide interoperability, scalability, and stability, it is not necessarily required in order to achieve a working solution. The listed IBM products can be integrated with the use of standard interfaces to existing services, components, and products from other vendors.

Note: Refer to Appendix A, “IBM software product descriptions” on page 433, for detailed descriptions of each IBM software product used in the Product mapping.

The following products are used in the provided Product mappings:

- Basic components:
  - IBM WebSphere Application Server (WAS)
  - IBM HTTP Server (IHS)
  - IBM Tivoli® Directory Server (TDS)
- Extension services and Client services:
  - IBM WebSphere Everyplace Deployment (WED)
  - IBM WebSphere Everyplace Access (WEA)
  - IBM WebSphere Everyplace Mobile Portal (WEMP)
  - IBM Workplace Client Technology (Micro Edition and Enterprise Edition, WCT-ME/EO)
  - IBM Workplace Managed Client (WMC)
- Solution options
  - IBM WebSphere Portal Server (WPS)
  - IBM WebSphere Voice Response (WVR)
  - IBM WebSphere Voice Server (WVS)
  - IBM Workplace Collaboration Server (WCS)
  - IBM Lotus Domino and Sametime® Everyplace
- Connectivity and Access for Mobile services
  - IBM WebSphere Connection Manager (WECM)

The following products are packaged in the products above:

- IBM DB2e (+Synchronization)
- IBM MQe (Embedded Messaging)
- IBM SMF - Service Management Framework (OSGi implementation)
- IBM J9 (the IBM Java Micro Edition Virtual Machine)
- IBM Tivoli Device Manager Server (DMS)

6.3.3 Description of the components in the component models

For convenience, this section summarizes the Distributed client services, Distributed client extension services, and Thin client extension services components used throughout the Runtime patterns and Product mappings. The reader can go through the following component descriptions or jump right into details found in Chapter 7, “Runtime patterns and Product mapping details” on page 155. The extension services we reference were described in 2.1, “Programming models” on page 28 (including the color coding).

Figure 6-3 shows a component model example. All high-level service components are included. They depend on the solution and vary by product.

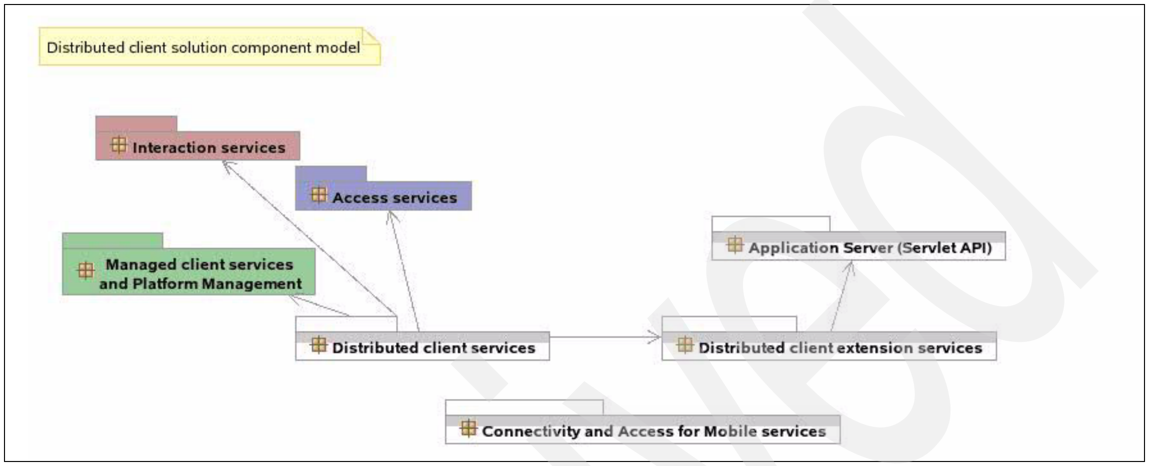

Distributed client solution component model
Interaction services
Access services
Managed client services and Platform Management
Application Server (Servlet API)
Distributed client services
Distributed client extension services
Connectivity and Access for Mobile services

Figure 6-3 Component model example for IBM WebSphere Everyplace Deployment V6.0

Client extension services

Figure 6-4 on page 145 provides an Analysis Model overview of those services that extend an existing service-oriented architecture to the edge (client device access capabilities). A description of the services can be found in Table 6-3 on page 145.

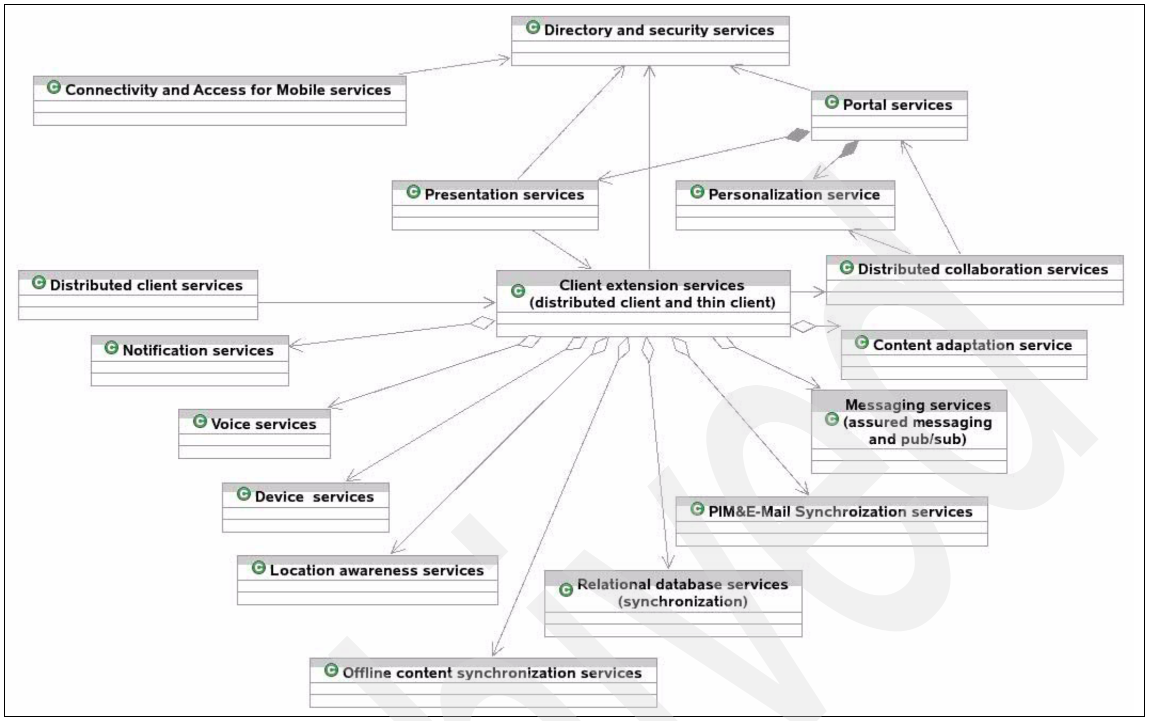

Directory and security services
Connectivity and Access for Mobile services
Portal services
Presentation services
Personalization service
Distributed client services
Client extension services (distributed client and thin client)
Distributed collaboration services
Notification services
Content adaptation service
Voice services
Messaging services (assured messaging and pub/sub)
Device services
PIM&E-Mail Synchroization services
Location awareness services
Relational database services (synchronization)
Offline content synchronization services

Figure 6-4 Services used to extend an SOA to the edge

Table 6-3 lists the server-side components that are included in the Thin and Distributed client extension, the Connectivity and Access for Mobile services, and the Distributed collaboration services. This table also contains product examples.

The corresponding client parts and additional client services components are summarized in Table 6-4 on page 151.

Table 6-3 Client extension and Distributed collaboration services components (server)

Extension services (server)	**Components**	**Description and product example**
Thin client extension services components (server)		
Portal services	Portal Server	Provides customized, personalized, and aggregated access to data and applications (also supports multi-user and role concepts), for example, IBM Portal Server.
Content Adaptation	Content Adaptation	Extends portal services with content adaptation for different devices and their browsers, for example, IBM WEMP[a] or WEA[b].
Distributed client extension services components (server)		

Extension services (server)	Components	Description and product example
Web Services	Web Services Client and Provider	Web Services client and provider implementations, for example, in IBM WebSphere Application Server (in WEA, [c], WEMP). There is also a client and provider in the WED client as well, though it operates based upon different JSR standards.
Relational database services (JDBC)	Database Synchronization Server	Service for synchronization of an (embedded) device database with a relational back-end database, for example, IBM DB2e SyncServer (part of WED and WEA).
Device services	Device Services Server	Provides device and Platform Management functions such as SW distribution, configuration, application control, and inventory, for example, IBM Tivoli Device Manager Server (in WED and WEA).
Assured messaging services (JMS)	Messaging Server	Server component for secure and reliable messaging (JMS), for example, IBM MQ Series (part of WED)
Pub/Sub messaging services	MQ Message Broker	Messaging broker capable of using a lightweight TCP/IP-based publish/subscribe protocol designed for low-bandwidth communication, for example, IBM MQ Series.
Offline content synchronization services	Offline Content Sync Servlet	Servlet that synchronizes static HTML pages and simple forms with a Web cache on the device, for example, IBM WebSphere Everyplace offline content sync client (available with WEA).
PIM/e-mail synchronization services	PIM/e-mail Sync Server	Synchronization of PIM[d] and e-mail data between Groupware server and device client (including offline availability), for example, IBM WEA (Everyplace SyncServer).
Notification services	Notification Server	Service that delivers triggered messages (alerts) over defined channels to clients to inform about certain states/state changes, for example, IBM INS[e] (part of WEA).
Location awareness services	Location Awareness Server	Common API to leverage information from multiple location service providers (provider transparency/differentiation and context-driven dynamic service binding).
Voice services	Voice Application Access	Extend applications with speech capabilities through VoiceXML markup, for example, IBM WebSphere Voice Application Access.
	Voice Server and Voice Response	Applications with speech capabilities (text-to-speech, automatic speech recognition), for example, IBM WebSphere Voice Server and Voice Response.
Distributed collaboration services components		

Extension services (server)	Components	Description and product example
Team collaboration services	Team Collaboration Services	Provides awareness, instant messaging, Web conferencing, and customizable team spaces, for example, IBM WCS[f] (might also be used as channel for notification services, for example, IBM Sametime Everyplace.)
Document services	Document Services	Delivers standards-based document management functions for rich and thin clients, for example, IBM WCS.
Messaging services	Messaging Services	Contains standards-based PIM and messaging capabilities like calendering and scheduling, personal address book, to-do lists, and e-mail for different clients, for example, IBM WCS. (Might also be used as channel for notification services.)
Web content management services	Web Content Management Services	Supports content creation and management process to enable author-once and publish-everywhere control, for example, IBM WCS.
Learning services	Learning Services	Provides a personalized, online learning environment (course discussion areas, document sharing, Web conferencing, chat rooms), for example, IBM WCS.
Connectivity and Access for Mobile services components (server)		
Secure remote connection services (Roaming, VPN)	Connectivity and Access for Mobile services	VPN gateway providing security, optimization, and session management for remote access connections, for example, IBM WECM[g].

a. WEMP - IBM WebSphere Everyplace Mobile Portal
b. WEA - IBM WebSphere Everyplace Access
c. WED - IBM WebSphere Everyplace Deployment
d. PIM - Personal Information Management (calendar, task list, address book, memo)
e. INS - IBM Intelligent Notification Server
f. WCS - IBM Workplace Collaboration Services
g. WECM - IBM WebSphere Everyplace Connection Manager

The following sections highlight the key components relevant to this book:

- IBM WebSphere Everyplace Mobile Portal V5.1 components
- IBM WebSphere Everyplace Deployment V6.0 components
- IBM WebSphere Everyplace Access V5.1 components
- IBM Workplace Collaboration Services V2.5.1 components

Note: Appendix A, “IBM software product descriptions” on page 433, includes additional information describing the IBM software products instantiated in the Product mappings.

IBM WebSphere Everyplace Mobile Portal V5.1 components

The IBM WebSphere Everyplace Mobile Portal (WEMP) components of interest are displayed in Figure 6-5. The XDIME Aggregator collects information about a user's request, selects related pages or portlets, and aggregates the output for display using XDIME. During the aggregation process, the aggregator evaluates the extended properties for each navigation node and portlet.

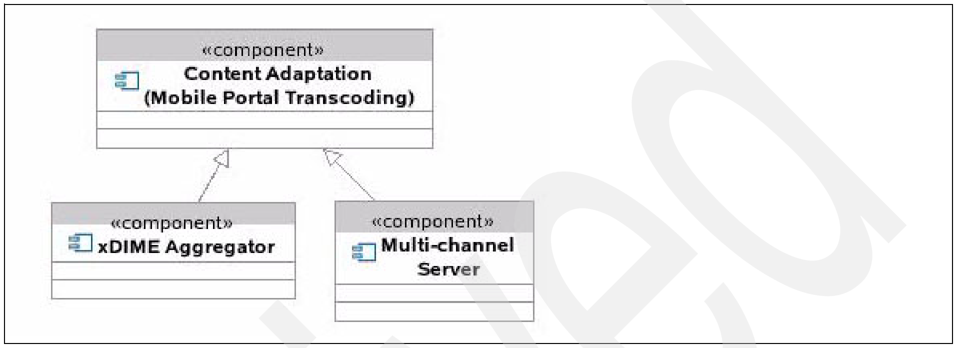


Figure 6-5 Thin client extension services components provided by IBM WebSphere Everyplace Mobile Portal V5.1 (WEMP)

The Multi-channel Server is the runtime component that processes the XDIME markup created by the XDIME Aggregator and XDIME portlets. Then it generates device-dependent markup and delivers this back to the Content Adaptation component for the final page composition and delivery.

IBM WebSphere Everyplace Deployment V6.0 components

IBM WebSphere Everyplace Deployment V6.0 (WED) is delivered with the following key components:

- Database Synchronization Server (DB2 Everyplace Sync Server)
- Device Services Server (Device Management Server)
- Messaging Server (MQ)

IBM WebSphere Everyplace Access V5.1 components

IBM WebSphere Everyplace Access V5.1 (WEA) consists of the following components:

- Database Synchronization Server (DB2 Everyplace Sync Server)
- Device Services Server (Device Management Server)
- Location Awareness Server
- (Portal) Content Adaptation
- Notification Server (Intelligent Notification Server - INS)
- Offline Content Sync Servlet
- PIM/e-mail Sync Server (TrueSync Server)

- Additionally available: Messaging Server (MQ)

The Database Synchronization Server, parts of the Device Services Server, and the Messaging Server are common for the IBM WebSphere Everyplace Deployment V6.0 product.

IBM Workplace Collaboration Services V2.5.1 components

IBM Workplace Collaboration Services is a completely integrated collaborative environment that includes a wide range of capabilities including:

- E-mail
- Calendaring and scheduling
- Awareness
- Instant messaging
- Learning
- Team spaces
- Web conferencing
- Document and Web content management.

With Workplace Collaboration Services, organizations can deploy a single collaboration solution for the enterprise right out of the box with the flexibility to deploy one, two, or more capabilities in any combination, all running within a single, fully integrated IBM Workplace collaboration environment.

Distributed client services (client)

The Distributed client services were introduced in 2.1, “Programming models” on page 28. They consist of *Managed client services and Platform Management, Access services*, and *Interaction services* (see Figure 6-3 on page 144 and Figure 6-6 on page 150). Table 6-4 on page 151 lists the Client services and the components that instantiate these services in the Product mappings of the Runtime pattern (see Chapter 7, “Runtime patterns and Product mapping details” on page 155).

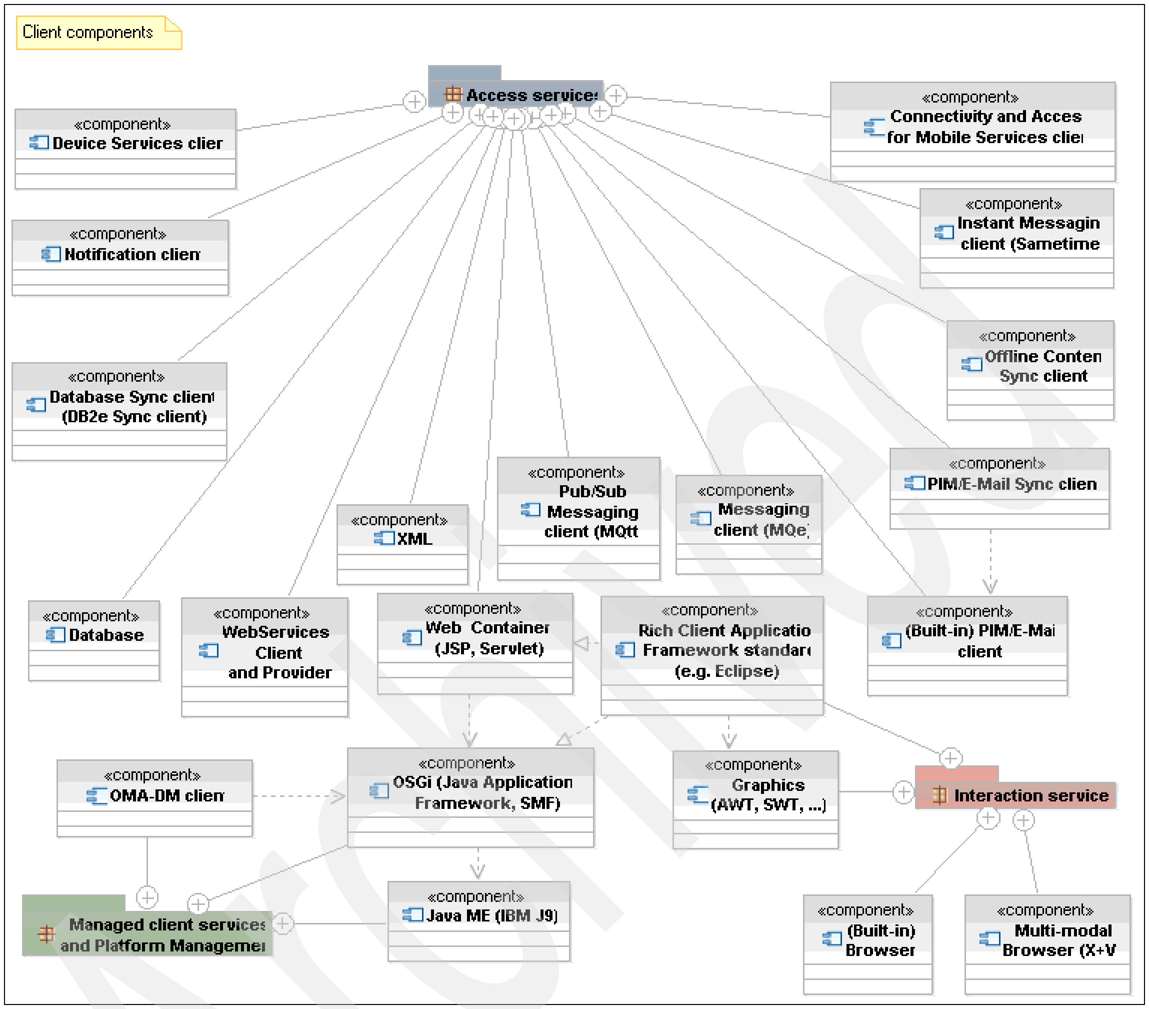

Client components
Access service:
«component» Device Services clier
«component» Notification clien
«component» Database Sync client (DB2e Sync client)
«component» Connectivity and Acces for Mobile Services clie
«component» Instant Messagin client (Sametime
«component» Offline Conten Sync client
«component» PIM/E-Mail Sync clien
«component» XML
«component» Pub/Sub Messaging client (MQtt
«component» Messaging client (MQe)
«component» Database
«component» WebServices Client and Provider
«component» Web Container (JSP, Servlet)
«component» Rich Client Applicatio Framework standar (e.g. Eclipse)
«component» (Built-in) PIM/E-Mai client
«component» OMA-DM clien
«component» OSGi (Java Application Framework, SMF)
«component» Graphics (AWT, SWT, ...)
Interaction service
Managed client services and Platform Manageme
«component» Java ME (IBM J9)
«component» (Built-in) Browser
«component» Multi-modal Browser (X+V

Figure 6-6 Client services and components

Figure 6-6 shows which components belong to which service. It also shows how components depend on each other and how components are part of others. All these components are available through IBM products. When an open standard was available, the implementations of this standard was considered (for example, the OSGi and Eclipse components). Those functions that are not available through standard interfaces are delivered by extensions of IBM products (for example, DB2e Sync client).

Table 6-4 Distributed client services and client components

Client services	**Components**	**Description and product example**
Managed client services and Platform Management		
Java runtime	Java ME[a]	Java Virtual Machine for devices with resource limitations (Java2 Micro edition standard), for example, IBM J9.
Java application framework	Eclipse	Eclipse RCP provides a standardized Java application framework with the capability for a local rich or Web-based UI.
Platform Management	OSGi	Platform management client is instantiated with the IBM Enterprise Management Agent (device agent), which extends the OSGi agent.
Access services		
Web Container	Web Container	WED 6.0 client supports JSP 2.0 and servlet 2.4. WCTME 5.7.2 supports JSP 1.2 and servlet 2.3.
XML Parsing services	XML	XML parser components for devices (MicroXML- small footprint non-validating parser, XML4J - full featured parser with validation), for example, provided with IBM [b] 6.0 client and WCTME-EO 5.8.1.
Secure remote connection services (Roaming, VPN)	Connectivity and Access for Mobile services client	Secure and connection optimized remote access client (VPN), for example, IBM WECM[c] client (Mobility client).
Offline content synchronization services	Offline Content Sync client	Provides static HTML pages and simple forms (XForms) offline on device and synchronizes with back-end original, for example, IBM WebSphere Everyplace offline content sync client (delivered with WEA[d]).
Notification services	Notification client	Client that is capable of receiving notifications (might also be able to kick off actions on the client in case of certain notifications), for example, built-in device (like SMS) or provided by a third party (for example, IBM Sametime client).
Assured messaging services (JMS)	Messaging client	Client component for secure and reliable messaging (JMS), for example, IBM WebSphere MQe.

Client services	Components	Description and product example
Pub/Sub messaging services	Pub/Sub Messaging client	Client using a lightweight TCP/IP-based publish/subscribe protocol designed for low-bandwidth communication with the MQ Message Broker, for example, IBM MQtt[e].
Web Services	Web Services Client and Provider	Web Services client (JSR 172 standard) and provider implementations, for example, in IBM WCT-ME[f].
Relational database services (JDBC)	Database	Small footprint of a relational database for devices with resource limitations, for example, IBM DB2e or IBM Cloudscape (WED, WCTME-EO).
	Database Sync client	Client of the database synchronization server, for example, IBM DB2e sync client (for example, in WED client, WCTME, WCTME-EO).
Device services	Device Services client	Client component for device services provided on operating system level, for example, IBM WEDM[g] clients (for example, provided with WEA).
Collaboration services: - Messaging services (PIM/e-mail)	PIM/e-mail Sync client	Synchronization of PIM[h] and e-mail data between Groupware server and device client (including offline availability), for example, IBM WEA sync client (Everyplace client) or standard OMA-DS client.
	(Built-in) PIM/e-mail client	Built-in the device or vendor-specific client for PIM and e-mail data, for example, IBM Lotus Notes® client, IBM Workplace Managed Client, Windows Mobile® client.
- Team collaboration services (Instant Messaging service)	Instant Messaging client	Instant messaging client component for instant messaging on the device platform (part of Team collaboration services). Might also be used as channel for notification services, for example, IBM Sametime client.
Interaction services		
Browser	(Built-in) Browser	Standard browser (HTML, WML) provided with the device and/or the device's operating system.
Multi-modal browser	Multi-modal Browser	Browser with support for (XHTML + VoiceXML (X+V), for example, Opera).

Client services	Components	Description and product example
Graphic services	Graphics	Provided standards like in Eclipse (AWT, SWT, lcdUI, others) or J2SE.
Composition of Client services		
Composition of Client services	Rich Client Application Framework standard	Standards that deliver a composition of Client services (components) in order to provide application management and enhanced user interface experience on the device, for example, Eclipse[i]. This can be extended with additional services, for example, collaboration service. Bundled with those, it is distributed as the IBM Workplace Managed Client.

a. Java ME - Java2 Micro edition (`http://java.sun.com`)
b. WED - IBM WebSphere Everyplace Deployment
c. WECM - IBM WebSphere Everyplace Connection Manager
d. WEA - IBM WebSphere Everyplace Access
e. MQtt - MQ telemetry transport
f. WCT-ME - Workplace Client Technology - Micro Edition
g. WEDM - IBM WebSphere Everyplace Device Manager
h. PIM - Personal Information Management (calendar, task list, address book, memo)
i. Eclipse - Standard for Java application platform with rich user interface capabilities (`http://www.eclipse.org`; also see the client programming models in 2.1, “Programming models” on page 28)

The listed components can be used alone or together. This depends on the user requirements and the environment. The following Runtime patterns and their Product mappings give an impression of which components can be used to fulfill a certain set of requirements.

7

Runtime patterns and Product mapping details

This chapter includes the Runtime pattern and corresponding Product mapping and component model details for each of the Application patterns.

Note: Refer to 6.3, "Runtime pattern nodes, products, and components" on page 137, for node and component descriptions, and Appendix A, "IBM software product descriptions" on page 433, for IBM product descriptions.

The chapter is organized into the following sections:

- Thin Client::Runtime pattern
- Thin Client::Runtime pattern=portal variation
- Voice-Enabled Client::Runtime pattern
- Distributed Presentation Client::Runtime pattern
- Distributed Application Client::Runtime pattern
- Distributed Rich Client::Runtime pattern
- Distributed Multimodal Client::Runtime pattern
- Distributed Collaboration Client::Runtime pattern
- Roaming Connectivity::Runtime pattern
- Composite::Runtime pattern

7.1 Thin Client::Runtime pattern

This section provides the details for the Runtime pattern and Product mapping for the Thin Client application pattern.

7.1.1 Runtime pattern

Figure 7-1 displays the Runtime pattern for the Thin Client application pattern. In this case, users with Web browser clients such as a desktop or laptop can access online (Web) applications. This Runtime pattern is representative of enterprise applications using the J2EE programming model.

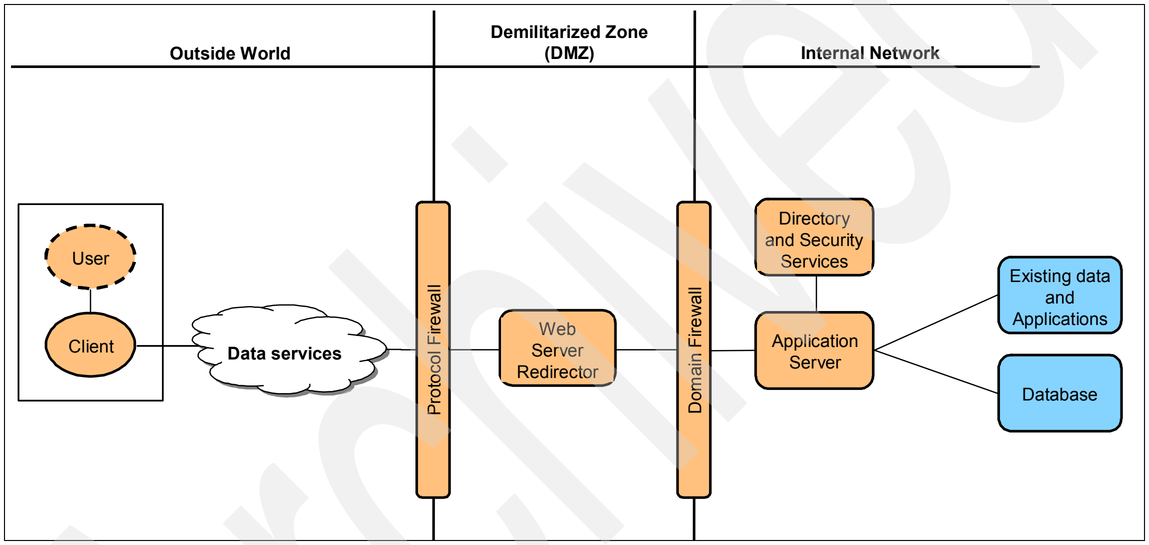


Figure 7-1 Thin Client::Runtime pattern for Web browser clients

Note: A description of the nodes can be found in 6.3, “Runtime pattern nodes, products, and components” on page 137.

The Web Server Redirector node delivers HTML content to the Client component, where it is rendered by the HTML Web browser. The page layout is designed for standard Web resolutions such as 1024x768, but can scale as standard within the browser for other resolutions such as 800x600 or 1280x1024.

A simple solution to support a limited number of additional devices with smaller screen sizes would be to provide device-dependent HTML pages. The user agent of the Web browser can be used to determine and activate the appropriate JSP for the application. The big drawback of this solution is that for each device with a smaller screen then the standard size will need a separate JSP, which will need to be developed and maintained. For example, a WML browser found in a WAP found will have a much smaller screen size than a standard PC-based Web browser client. In order to keep the development effort for the View components to a minimum, the variety of devices with which the application is accessed should be limited. This is only practical if an enterprise can define the type of device that will access the enterprise application, thus reducing the number of pages that must be customized and maintained.

7.1.2 Product mapping

Figure 7-2 displays the Product mapping for the Thin Client::Runtime pattern. The primary IBM software products for the Product mapping are the IBM HTTP Server and WebSphere Application Server, which are available on a variety of operating system platforms (Windows 2000/2003, AIX®, Linux, Solaris™).

The IBM HTTP Server and WebSphere plug-in on the Web Server Redirector node deliver the requested content to the device. The WebSphere Application Server provides the enterprise application runtime and access to back-end data and/or applications.

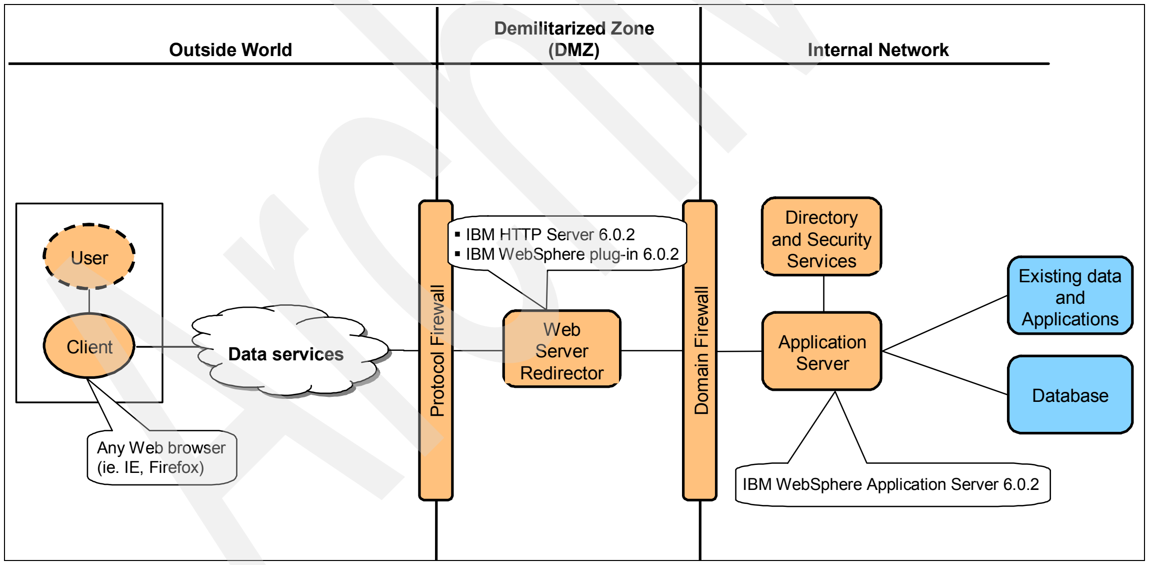


Figure 7-2 Thin Client::Runtime pattern::Product mapping

Note: Refer to Appendix A, “IBM software product descriptions” on page 433, for information about the IBM products instantiated in the Product mapping.

7.2 Thin Client::Runtime pattern=portal variation

This section provides the details for the Runtime pattern=portal variation, Product mapping, and component model for the Thin Client application pattern.

7.2.1 Runtime pattern

Figure 7-3 displays a portal-based variation of the Thin Client runtime pattern. There are two key drivers that distinguish the need for using the portal variation of the Thin Client runtime pattern:

- Access to role-based and/or personalized aggregated content
- Content adaptation and rendering for different device types

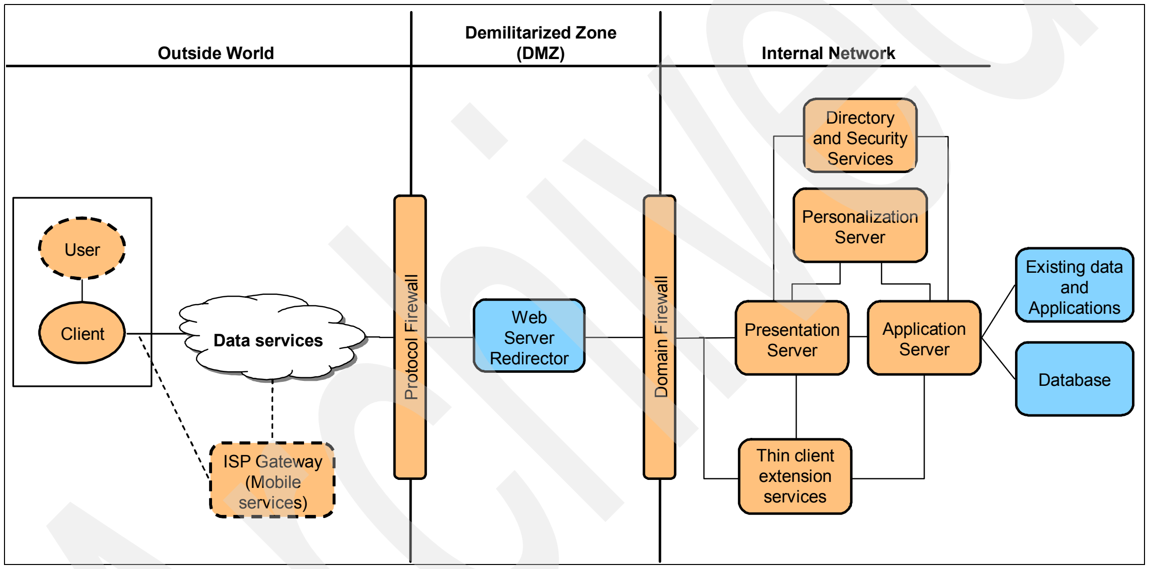


Figure 7-3 Thin Client::Runtime pattern=portal variation

The demand to access information from less sophisticated devices is increasing. As seen in Figure 7-3, the Thin client extension services provide Web browser access to a wide range of devices, including high function devices such as desktops and laptops, as well as lower function PDAs, mobile phones, and RFID readers.

The Thin client extension services extend the existing Presentation services with multi-device support for online access (for example, to a Portal). They are preferably implemented in an Application server environment. The extension services might also provide multi device online access to collaboration services.

Note that no specific services on the client (device) side are required to achieve this so that very thin clients (low function devices) can be used. Depending on the capabilities of the device, the Thin client::Runtime pattern might be combined with the Roaming Connectivity runtime pattern to gain secure managed access to the corporate network (see 7.9, “Roaming Connectivity::Runtime pattern” on page 189).

7.2.2 Product mapping

Figure 7-4 shows the Thin Client::Runtime pattern=portal variation::Product mapping. There are two new IBM products to highlight in Figure 7-4. The IBM WebSphere Portal Server (WPS) V5.1 provides the ability to aggregate and personalize content.

The IBM WebSphere Everyplace Mobile Portal (WEMP) V5.1 uses XDime technology, which is a device-independent mark-up language, to provide similar support as WebSphere Portal for delivering rich content to mobile devices. WEMP provides a means of supporting many device types (including low function mobile devices), without the need to manually create and maintain customized JSPs for each device.

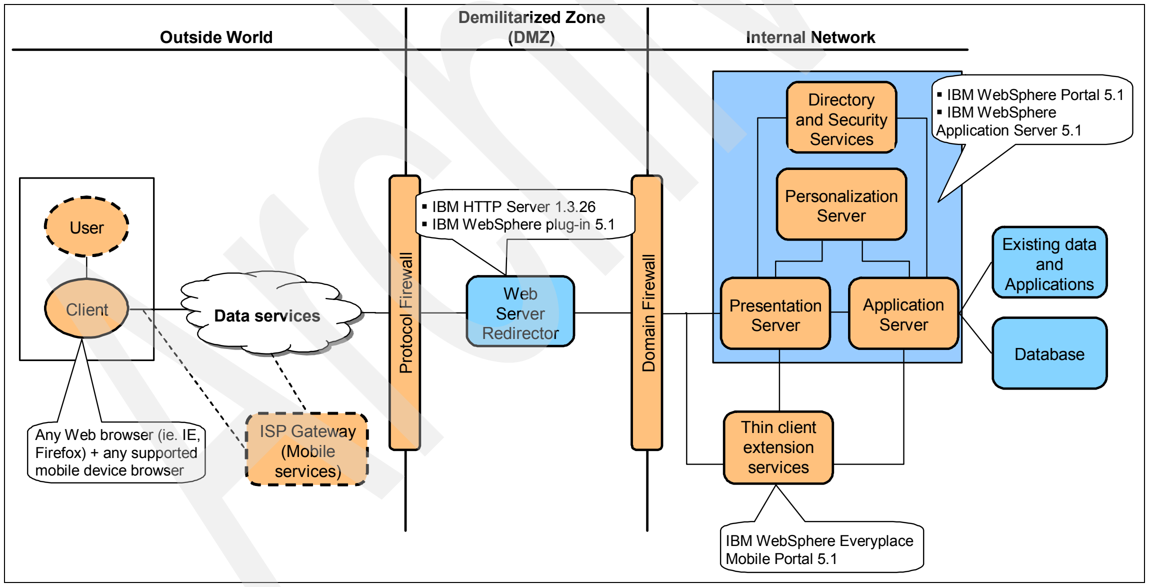

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Any Web browser (ie. IE, Firefox) + any supported mobile device browser
Data services
ISP Gateway (Mobile services)
Protocol Firewall
▪ IBM HTTP Server 1.3.26
▪ IBM WebSphere plug-in 5.1
Web Server Redirector
Domain Firewall
Directory and Security Services
▪ IBM WebSphere Portal 5.1
▪ IBM WebSphere Application Server 5.1
Personalization Server
Presentation Server
Application Server
Existing data and Applications
Database
Thin client extension services
IBM WebSphere Everyplace Mobile Portal 5.1

Figure 7-4 Thin Client::Runtime pattern=portal variation::Product mapping

7.2.3 Component model

Figure 7-5 shows the architecture components (services) for the Thin client extension services node of the Thin Client::Runtime pattern= portal variation=Product mapping (IBM software products).

The Thin client extension services are provided by the IBM WebSphere Everyplace Mobile Portal (WEMP), which is based on IBM WebSphere Portal server (WPS). The WebSphere Portal Server runs on top of the IBM WebSphere Application Server environment, which provides for security and scalability.

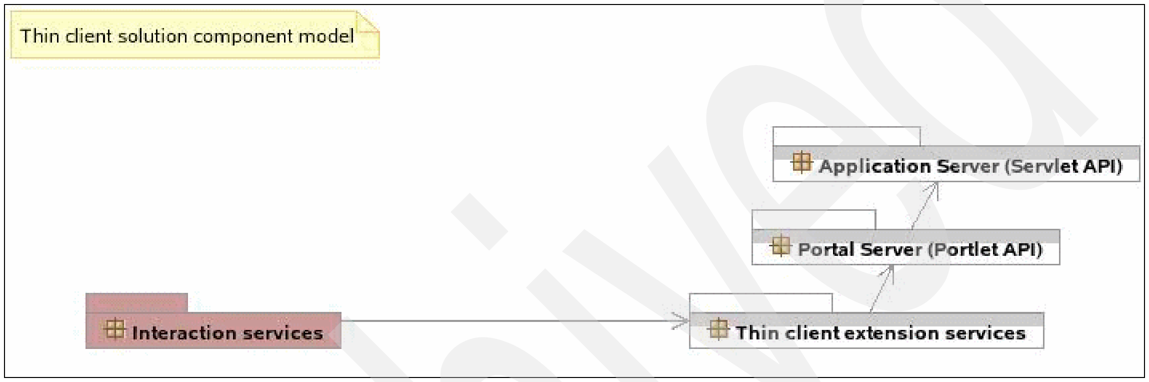

Thin client solution component model
Application Server (Servlet API)
Portal Server (Portlet API)
Interaction services
Thin client extension services

Figure 7-5 Architecture components of WebSphere Everyplace Mobile Portal (WEMP) V5.1 solutions

Figure 7-6 shows the minimum services provided by the Interaction services for the Thin Client. Only a simple browser (WML, HTML, cHTML, etc.) is needed and typically built in to the device or provided by the device's operating system.

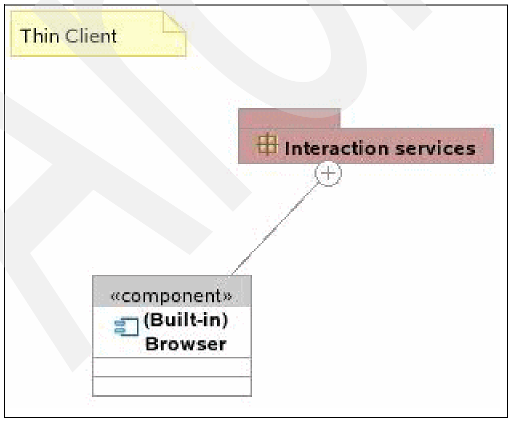


Figure 7-6 Thin client services components

7.3 Distributed Presentation Client::Runtime pattern

This section provides the details for the Runtime pattern, Product mapping, and component model for the Distributed Presentation Client application pattern.

7.3.1 Runtime pattern

The Distributed Presentation Client::Runtime pattern is portal based. It extends the Thin Client::Runtime pattern=portal variation with support for disconnectable operation of the client. This implies that the client must have local services in order to support these off-line capabilities. The *Distributed client services (presn)* node in Figure 7-7 provides these capabilities. This node works together with the *Distributed client extension services* node to ensure off-line availability of data on the client as well as update capabilities with back-end data and/or applications (synchronization).

Note: This pattern only provides the presentation service on the client. Application-dependent logic must be provided by the application developer.

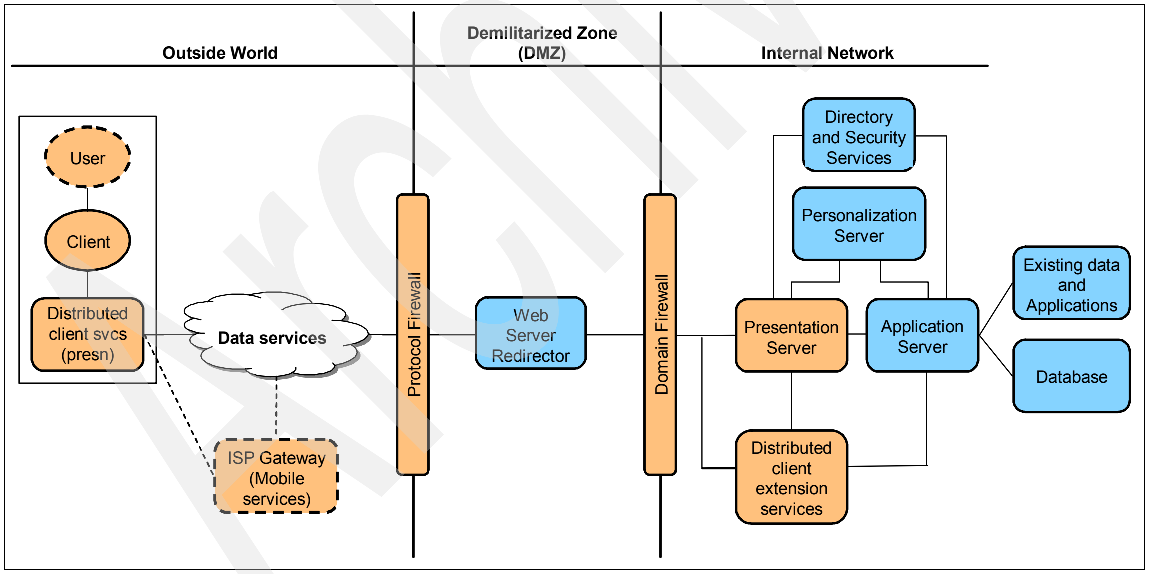


Figure 7-7 Distributed Presentation Client::Runtime pattern

7.3.2 Product mapping

IBM WebSphere Everyplace Access (WEA) V5.1 is used to instantiate the Distributed client services and Distributed client extension services nodes through its server and client components (see Figure 7-8). The server components run on IBM WebSphere Application Server V5.1. IBM WebSphere Portal V5.1 services are used for configuration, administration, and content adaptation.

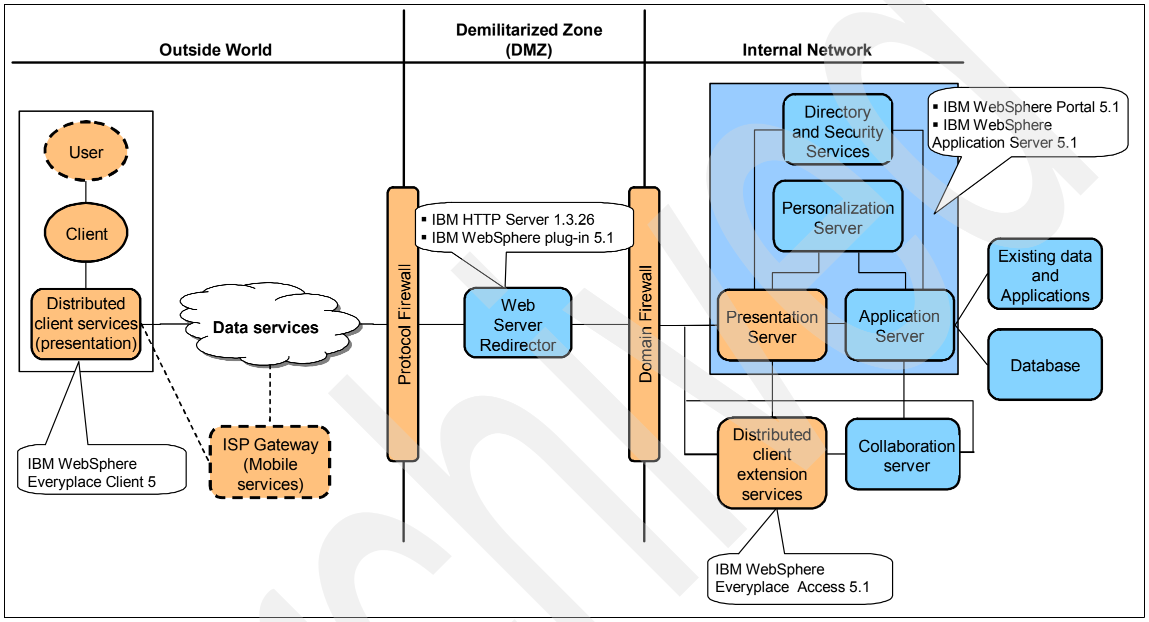

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (presentation)
IBM WebSphere Everyplace Client 5
Data services
ISP Gateway (Mobile services)
Protocol Firewall
▪ IBM HTTP Server 1.3.26
▪ IBM WebSphere plug-in 5.1
Web Server Redirector
Domain Firewall
Directory and Security Services
▪ IBM WebSphere Portal 5.1
▪ IBM WebSphere Application Server 5.1
Personalization Server
Presentation Server
Application Server
Existing data and Applications
Database
Distributed client extension services
Collaboration server
IBM WebSphere Everyplace Access 5.1

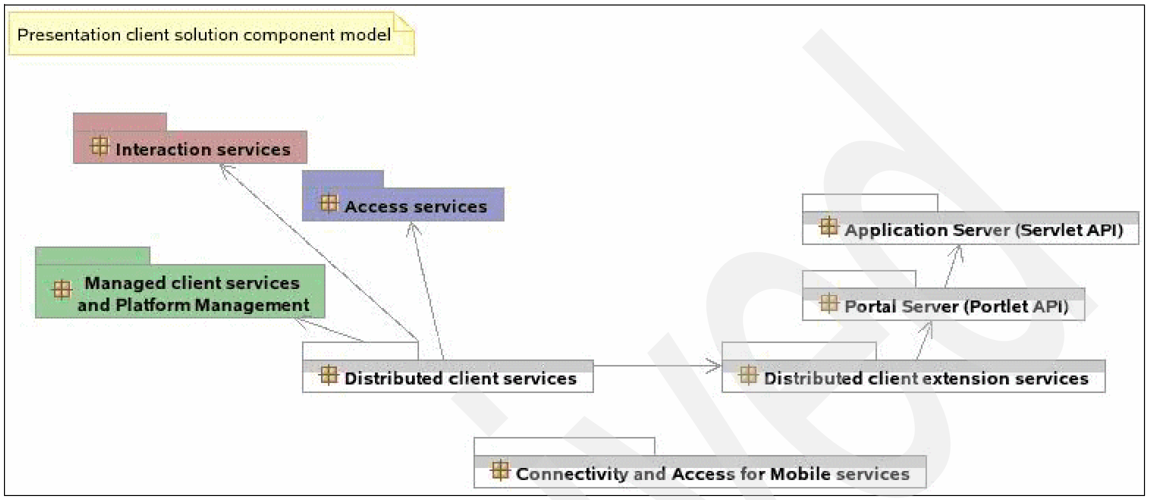
Figure 7-8 Distributed Presentation Client::Runtime pattern::Product mapping

IBM WebSphere Everyplace Access V5.1 provides components that can instantiate all or subsets of the following Distributed client extension services:

- Portal Services (extensions)
- Content Adaptation (extensions)
- Web Services
- Relational Database Services
- Device Services
- Offline Content Synchronization Services
- PIM/e-mail Synchronization Services
- Notification Services
- Location Awareness Services

7.3.3 Component model

The architectural components and services of the Distributed Presentation Client::Runtime pattern are displayed in Figure 7-9.



Presentation client solution component model
Interaction services
Access services
Managed client services and Platform Management
Distributed client services
Application Server (Servlet API)
Portal Server (Portlet API)
Distributed client extension services
Connectivity and Access for Mobile services

Figure 7-9 Architecture components of WebSphere Everyplace Access V5.1 solutions

The following components from Table 6-3 on page 145 are used to instantiate the Distributed client extension services:

- Content adaptation (extensions to Portal Server)
- Database synchronization server
- Device services server
- Offline content sync servlet
- PIM/e-mail sync server
- Intelligent notification server
- Location awareness server

Figure 7-10 on page 164 shows the components on the client side (device), which deliver the required Client services. The Client services that are required for this Runtime pattern belong to the Access and Interaction services. The Access services are delivered by the client product components of IBM WebSphere Everyplace Access. For local data access on the device, built-in device user interfaces and applications are used (for example, browser, built-in PIM/e-mail client).

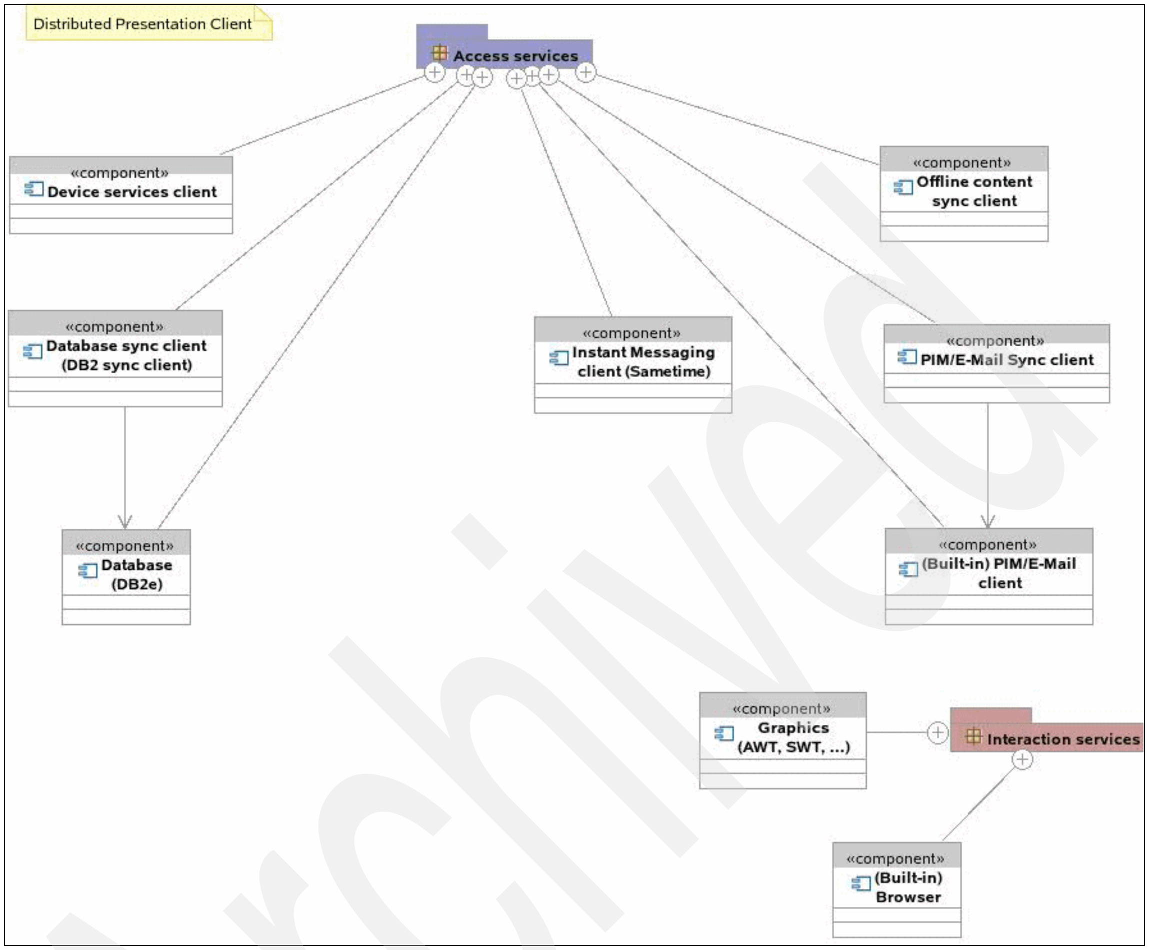

Distributed Presentation Client
Access services
«component» Device services client
«component» Database sync client (DB2 sync client)
«component» Database (DB2e)
«component» Instant Messaging client (Sametime)
«component» Offline content sync client
«component» PIM/E-Mail Sync client
«component» (Built-in) PIM/E-Mail client
«component» Graphics (AWT, SWT, ...)
Interaction services
«component» (Built-in) Browser

Figure 7-10 Distributed (presentation) client services components (WEA5.1 client)

7.4 Distributed Application Client::Runtime pattern

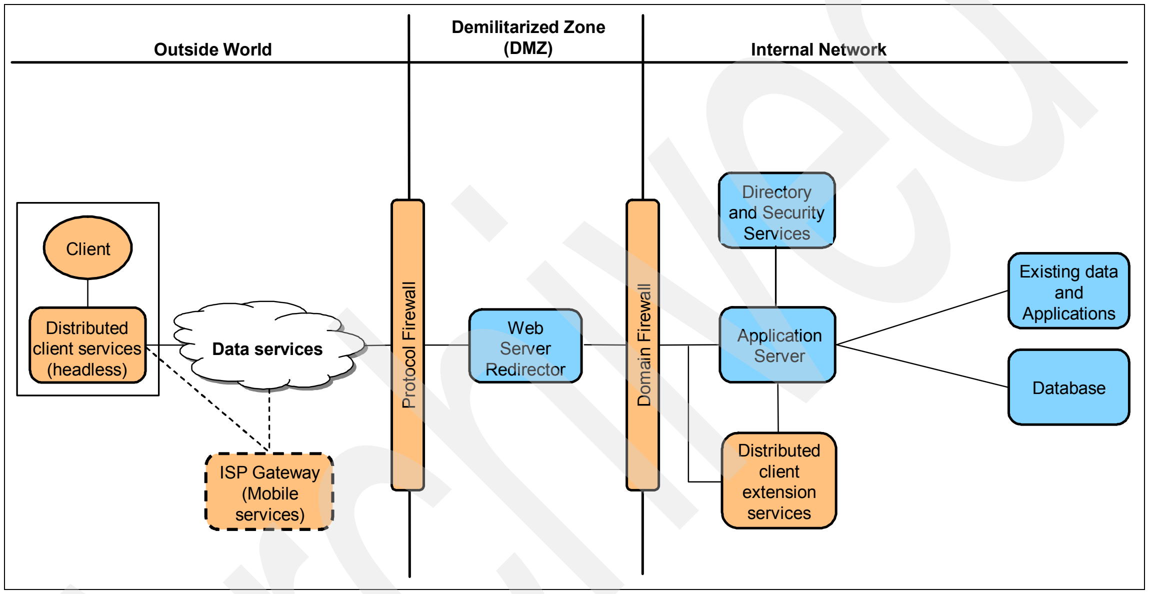
This section provides the details for the Runtime pattern, Product mapping, and component model for the Distributed Application Client application pattern.

7.4.1 Runtime pattern

The Distributed Application Client::Runtime pattern displayed in Figure 7-11 on page 165 is well suited for server-managed client applications without local user interface components (known as headless).

The meaning of distributed application in this context addresses the fact that a custom application is running on a (remote) device where user interaction for application management is not possible. Good examples for this type of applications are *embedded applications* (for example, in cars, home gateways, set-top boxes, RFID readers).

The Distributed client services (headless) node and its corresponding server part, the Distributed client extension services node, provide the underlying services for the integration of such a remote application (and device) into back-end business processes.



Outside World
Demilitarized Zone (DMZ)
Internal Network
Client
Distributed client services (headless)
Data services
ISP Gateway (Mobile services)
Protocol Firewall
Web Server Redirector
Domain Firewall
Directory and Security Services
Application Server
Distributed client extension services
Existing data and Applications
Database

Figure 7-11 Distributed Application Client::Runtime pattern (headless application)

The following services from the Distributed client extension services node (see Table 6-4 on page 151) are used:

- Relational database services
- Assured messaging services
- Device services
- Web Services

The Distributed client services contain the following counterparts to the server services and the additional services (see Table 6-4 on page 151):

- Managed client services and Platform Management:
  - Java runtime
  - Java application framework

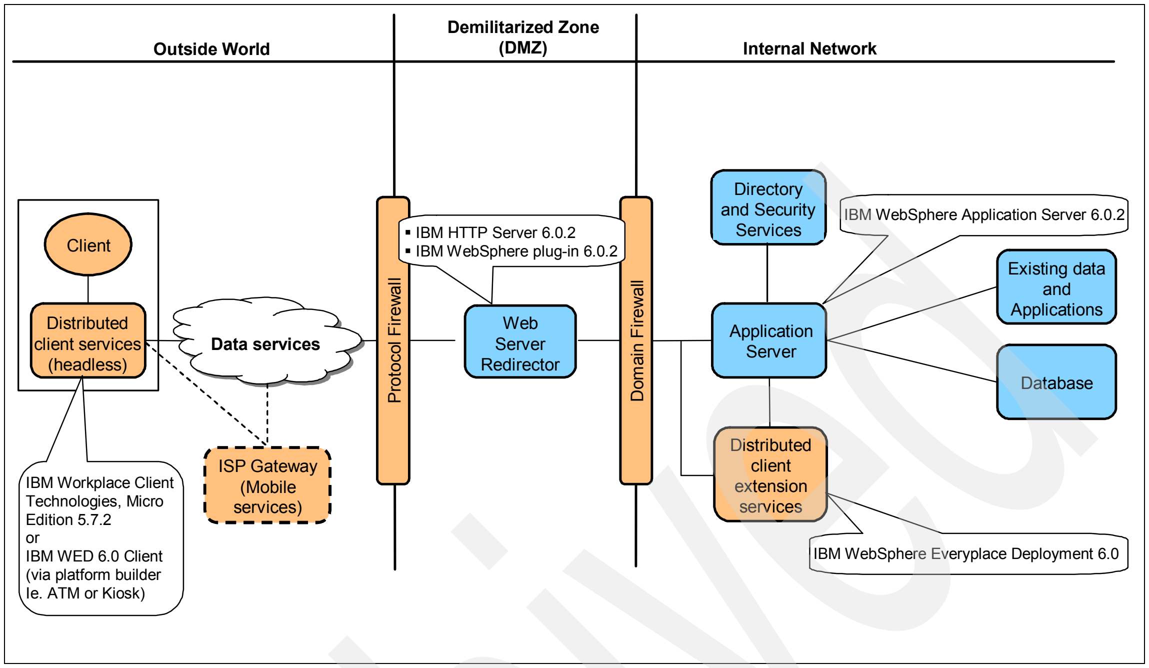
  - Platform Management
- Access services
  - Web Container
  - XML Parsing services
  - Assured messaging services
  - Pub/Sub messaging services
  - Web Services
  - Relational database services
  - Device services

The Managed client services and Platform Management are the heart of this solution on the device. They provide an open standard Java application platform (OSGi based) for extending the client capabilities to address some of the resource limitations of embedded devices. They also enable remote management of these Java applications running in the OSGi framework (bundles) through standard interfaces (OMA DM). The additional Access services provide (sub-sets) of the J2EE server-side services to allow a synchronous (common) programming model on the device and the server.

7.4.2 Product mapping

The Distributed client services and Distributed client extension services are instantiated by the IBM WebSphere Everyplace Deployment V6.0 (server) and IBM Workplace Client Technology, Micro Edition V5.7.2 (client).

IBM Workplace Client Technology, Micro Edition (WCTME) contains the components for the Client services, as seen in Figure 7-14 on page 169. The Client services (Managed client services, Platform Management, and Access services) provide a synchronous (common) programming model and seamless back-end integration. The IBM WebSphere Everyplace Deployment server provides the corresponding services to the client services for the back-end integration.



Outside World
Demilitarized Zone (DMZ)
Internal Network
Client
Distributed client services (headless)
Data services
ISP Gateway (Mobile services)
Protocol Firewall
IBM HTTP Server 6.0.2
IBM WebSphere plug-in 6.0.2
Web Server Redirector
Domain Firewall
Directory and Security Services
IBM WebSphere Application Server 6.0.2
Application Server
Existing data and Applications
Database
Distributed client extension services
IBM WebSphere Everyplace Deployment 6.0
IBM Workplace Client Technologies, Micro Edition 5.7.2 or IBM WED 6.0 Client (via platform builder Ie. ATM or Kiosk)

Figure 7-12 Distributed Application Client::Runtime pattern::Product mapping (headless application)

7.4.3 Component model

Figure 7-13 on page 168 displays the components provided by IBM WebSphere Everyplace Deployment V6.0 and IBM Workplace Client Technology, Micro Edition V5.7.2. The client services are classified as follows:

- Interaction services
- Access services
- Managed clients and Platform Management services

Since the Distributed application client does not provide a user interface, the local Interaction services are not deployed.

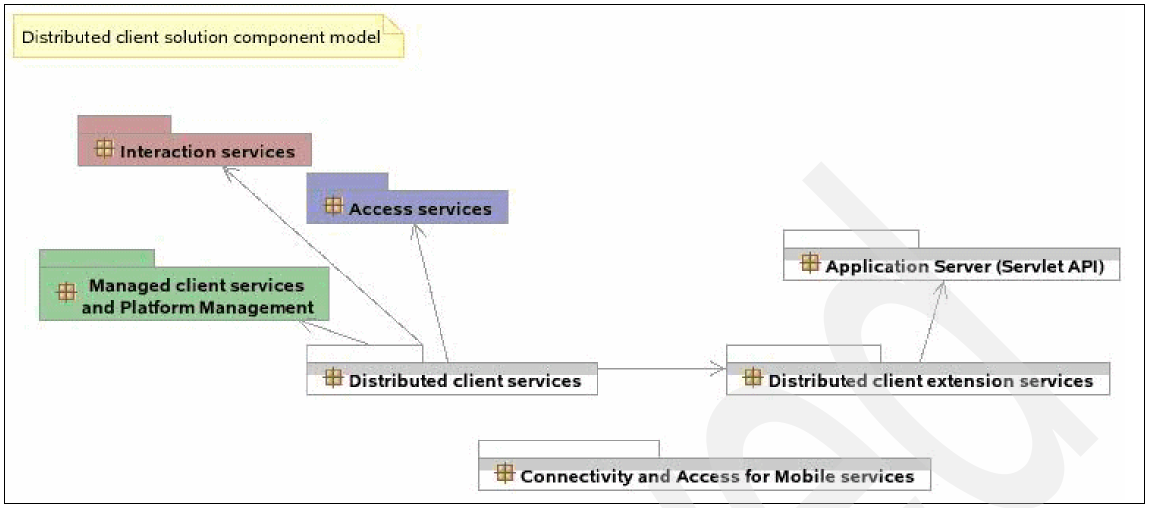

Distributed client solution component model
Interaction services
Access services
Managed client services and Platform Management
Application Server (Servlet API)
Distributed client services
Distributed client extension services
Connectivity and Access for Mobile services

Figure 7-13 Architecture components of WebSphere Everyplace Deployment V6.0 solutions

The following components of the IBM WebSphere Everyplace Deployment V6.0 server instantiate the Distributed client extension services:

- ► Database Synchronization Server
- ► Device Services Server
- ► Messaging Server

Figure 7-14 on page 169 displays the Client services components. The Interaction services were not included for this Runtime pattern, since they are not used.

The following components are available on the client:

- ► Managed client services and Platform Management:
  - – Java ME (IBM J9)
  - – OSGi (IBM SMF)
  - – OMA DM client (IBM DM agent for OSGi)
- ► Access services:
  - – Web Container (provided with IBM SMF)
  - – XML
  - – Messaging client (IBM MQe)
  - – Pub/Sub Messaging client (IBM MQ)
  - – Web Services Client and Provider
  - – Database (IBM DB2e Sync Server)
  - – Database Sync client (DB2e synchronization client)
  - – Device Services client (IBM Enterprise Management Agent)

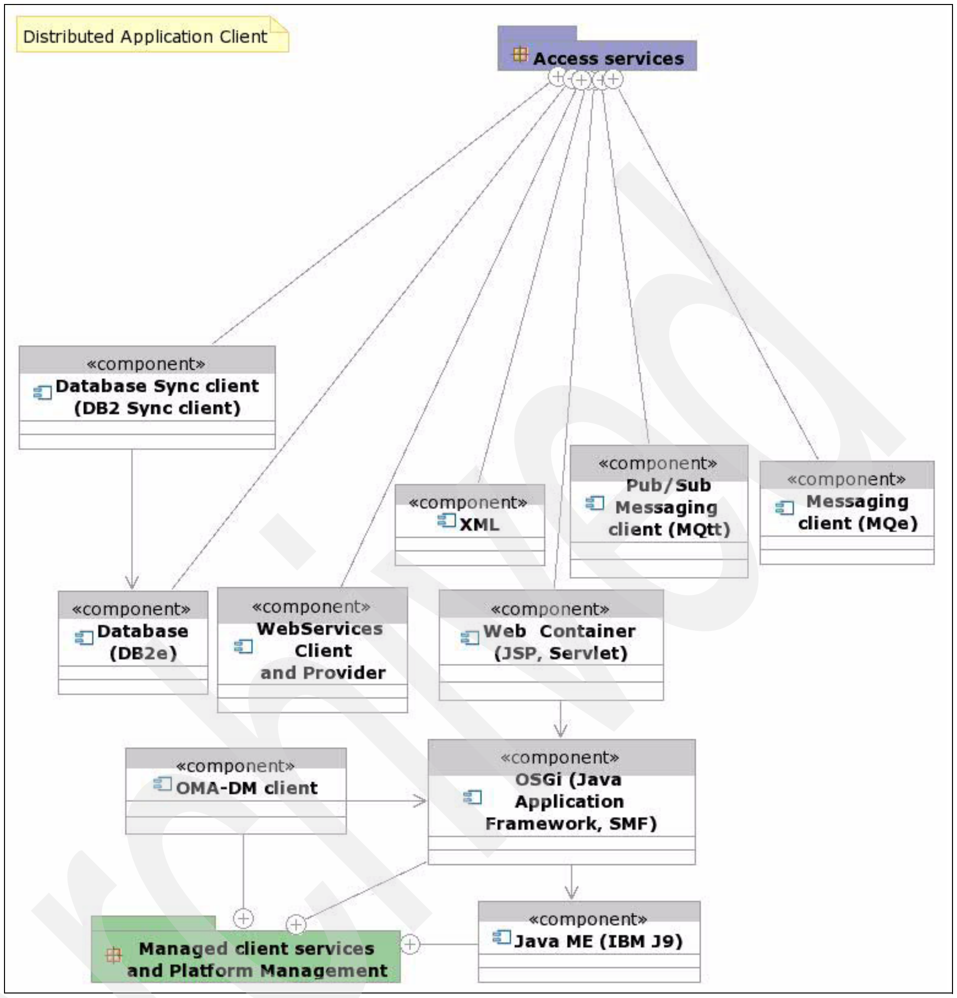

Distributed Application Client
Access services
«component» Database Sync client (DB2 Sync client)
«component» XML
«component» Pub/Sub Messaging client (MQtt)
«component» Messaging client (MQe)
«component» Database (DB2e)
«component» WebServices Client and Provider
«component» Web Container (JSP, Servlet)
«component» OMA-DM client
«component» OSGi (Java Application Framework, SMF)
«component» Java ME (IBM J9)
Managed client services and Platform Management

Figure 7-14 Distributed (application) client services components (headless application)

7.5 Distributed Rich Client::Runtime pattern

This section provides the details for the Runtime pattern and Product mapping for the Distributed Rich Client application pattern.

7.5.1 Runtime pattern

The Distributed Rich Client::Runtime pattern extends the DIstributed Application Client::Runtime pattern with support for a user interface on the device (see

Figure 7-15). The Distributed Rich Client runtime pattern provides Distributed client extension services and Client services that are required to support disconnectable applications using synchronous and asynchronous communication to back-end servers.

The addition of a user interface running on the local device requires greater resources on the device. Some devices are not able to run user interface standards like SWT or do not have a built-in microphone and speakers for multimodal access. Most devices are capable of providing a Web user interface in a browser. These additional services belong to the Interaction services component (for example, built-in browser or AWT provided with Eclipse).

There are two client device classes for this Runtime pattern:

- Full function devices (that is, laptop)

  The combination of the full function device and Distributed client extension services allows for a Rich or Web user interface on the local device. The Eclipse standard, which also uses OSGi, is proposed as the Java application platform on the device in order to provide a J2EE-like programming model.

- High function devices (that is, PDA)

  The high function devices are more limited than full function devices. The Distributed client extension services in this case support a Web-based UI, and local UI using SWT and AWT directly without the aggregation capabilities available in Eclipse.

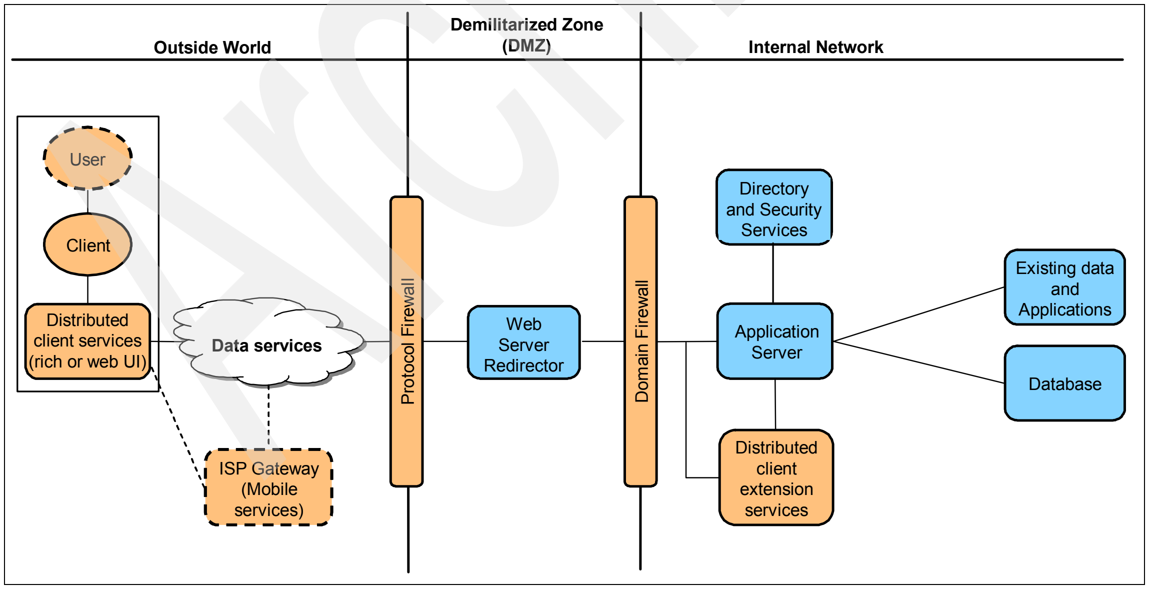

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (rich or web UI)
Data services
ISP Gateway (Mobile services)
Protocol Firewall
Web Server Redirector
Domain Firewall
Directory and Security Services
Application Server
Distributed client extension services
Existing data and Applications
Database

Figure 7-15 Distributed Rich Client::Runtime pattern

The Distributed client extension services provide the same function as in the Distributed Application Client::Runtime pattern:

- Relational database services
- Assured transactional messaging services
- Device services

The Distributed client services contain the following counterparts to the server services and the additional services (see Table 6-4 on page 151):

- Managed client services and Platform Management:
  - Java runtime
  - Java application framework
  - Platform Management
- Access services
  - Web Container
  - XML parsing services
  - Pub/Sub messaging services
  - Web services
- Interaction services
  - Browser
  - Graphic services

Eclipse provides the Managed client services, Platform Management, and Interaction services (OSGi), and part of the Access services. Eclipse is a Java application platform with OSGi (runtime environment). SWT, AWT, and Swing are used to develop applications with a rich UI. Eclipse provides components (including SWT and JFace) that enable aggregation.

7.5.2 Distributed Rich Client::Runtime pattern::Product mapping

The Distributed client services and Distributed client extension services are instantiated by the following IBM products for this pattern (see Figure 7-16 on page 172):

- Server:
  - IBM WebSphere Everyplace Deployment V6.0
- Clients:
  - IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux

  or

  - IBM Workplace Client Technology Micro Edition, Enterprise Offering V5.8.1 (WCTME-EO)

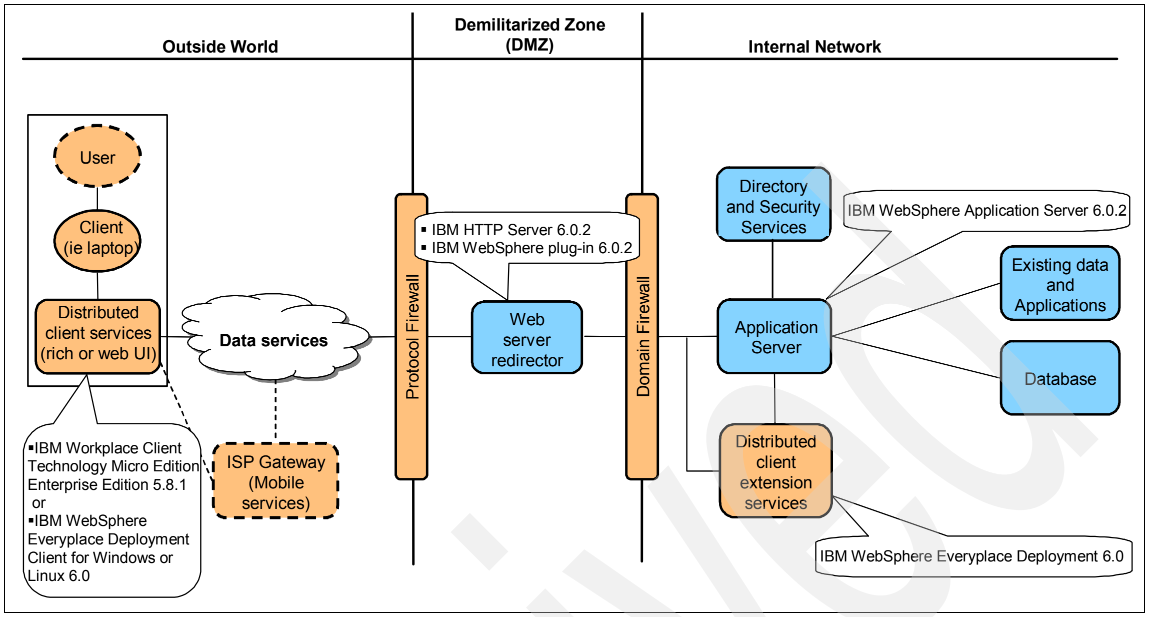

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client (ie laptop)
Distributed client services (rich or web UI)
Data services
ISP Gateway (Mobile services)
▪IBM Workplace Client Technology Micro Edition Enterprise Edition 5.8.1 or
▪IBM WebSphere Everyplace Deployment Client for Windows or Linux 6.0
Protocol Firewall
▪ IBM HTTP Server 6.0.2
▪ IBM WebSphere plug-in 6.0.2
Web server redirector
Domain Firewall
Directory and Security Services
IBM WebSphere Application Server 6.0.2
Application Server
Existing data and Applications
Database
Distributed client extension services
IBM WebSphere Everyplace Deployment 6.0

Figure 7-16 Distributed Rich Client::Runtime pattern::Product mapping

7.5.3 Distributed Rich Client::Runtime pattern::Product mapping=non PC form factor variation

The Distributed client extension services in this case support a Web-based UI and local UI using SWT and AWT directly without the aggregation capabilities available in Eclipse. This Product mapping is targeted at non-PC form factor devices. Devices of this type typically have a simple user interface found in high function devices such as a PDA.

This Product mapping removes the advanced graphical support (libraries) and reduces the user interface to a Web browser. All other services and nodes are identical to the DIstributed Rich Client::Runtime pattern.

The Distributed client services and Distributed client extension services are instantiated by the following IBM products (see Figure 7-17 on page 173):

- IBM WebSphere Everyplace Deployment V6.0 (server)
- IBM Workplace Client Technology Micro Edition V5.7.2 (client)

The client products are based on the Java2 Micro Edition and the OSGi standard. They also contain additional services to enable a common synchronous programming model for the server and the client, and seamless back-end integration (see Figure 7-17 on page 173).

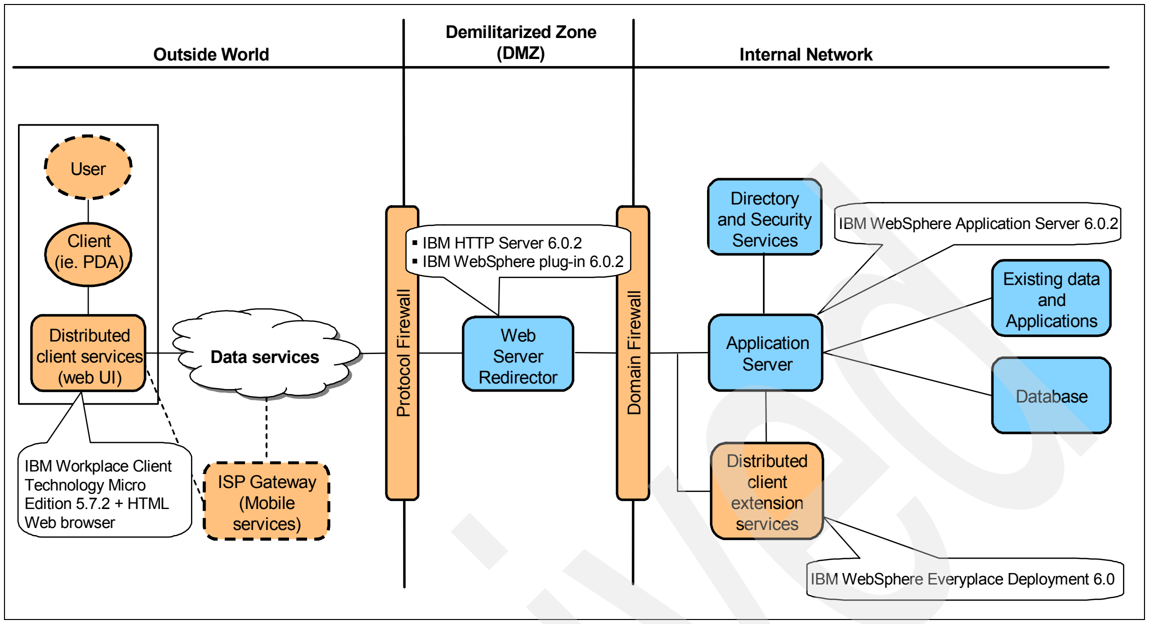

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client (ie. PDA)
Distributed client services (web UI)
Data services
IBM Workplace Client Technology Micro Edition 5.7.2 + HTML Web browser
ISP Gateway (Mobile services)
Protocol Firewall
▪ IBM HTTP Server 6.0.2
▪ IBM WebSphere plug-in 6.0.2
Web Server Redirector
Domain Firewall
Directory and Security Services
IBM WebSphere Application Server 6.0.2
Application Server
Existing data and Applications
Database
Distributed client extension services
IBM WebSphere Everyplace Deployment 6.0

Figure 7-17 Distributed Rich Client::Runtime pattern::Product mapping=non PC form factor variation

7.5.4 Component model

Figure 7-18 on page 174 displays the components of the Distributed Rich Client::Runtime pattern::Product mapping.

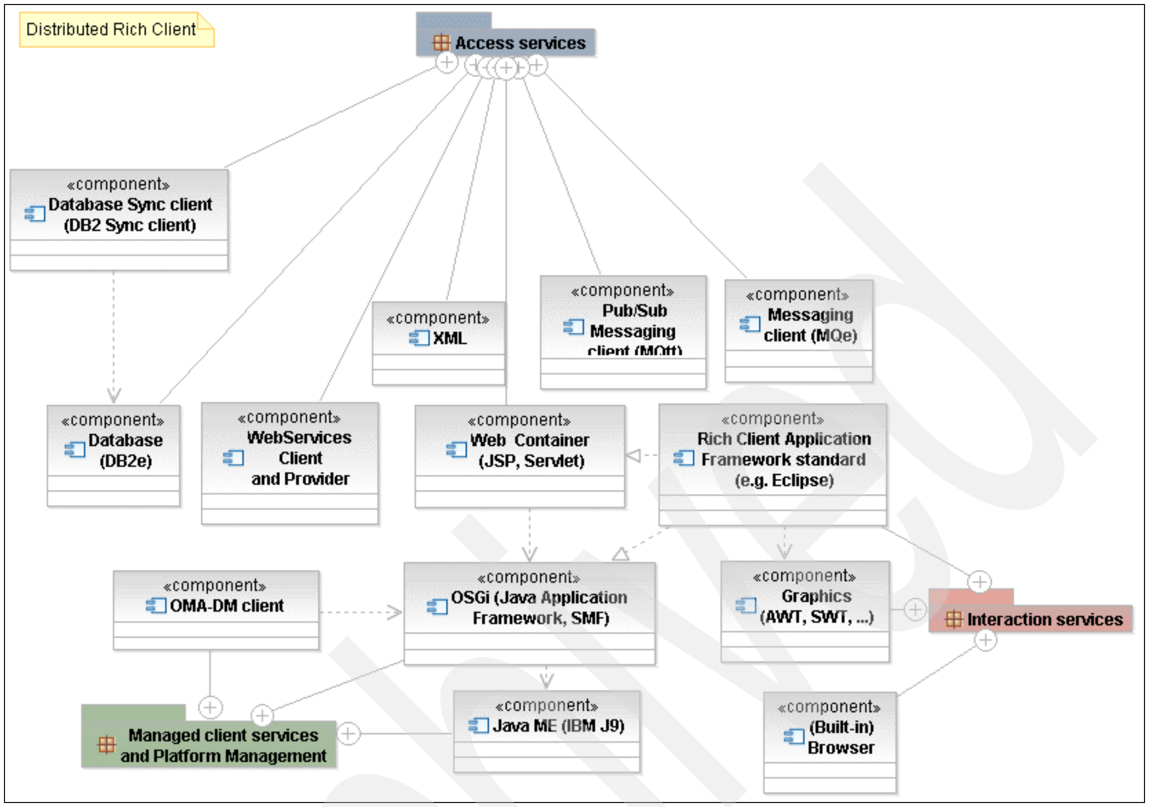

Distributed Rich Client
Access services
«component» Database Sync client (DB2 Sync client)
«component» XML
«component» Pub/Sub Messaging client (MQtt)
«component» Messaging client (MQe)
«component» Database (DB2e)
«component» WebServices Client and Provider
«component» Web Container (JSP, Servlet)
«component» Rich Client Application Framework standard (e.g. Eclipse)
«component» OMA-DM client
«component» OSGi (Java Application Framework, SMF)
«component» Graphics (AWT, SWT, ...)
Interaction services
Managed client services and Platform Management
«component» Java ME (IBM J9)
«component» (Built-in) Browser

Figure 7-18 Component model for the Distributed Rich Client

Figure 7-19 on page 175 shows the Client services components for the non-PC form factor variation. Components of the Interaction services are reduced to the built-in browser, since only simple user interface capabilities are needed for this Runtime pattern.

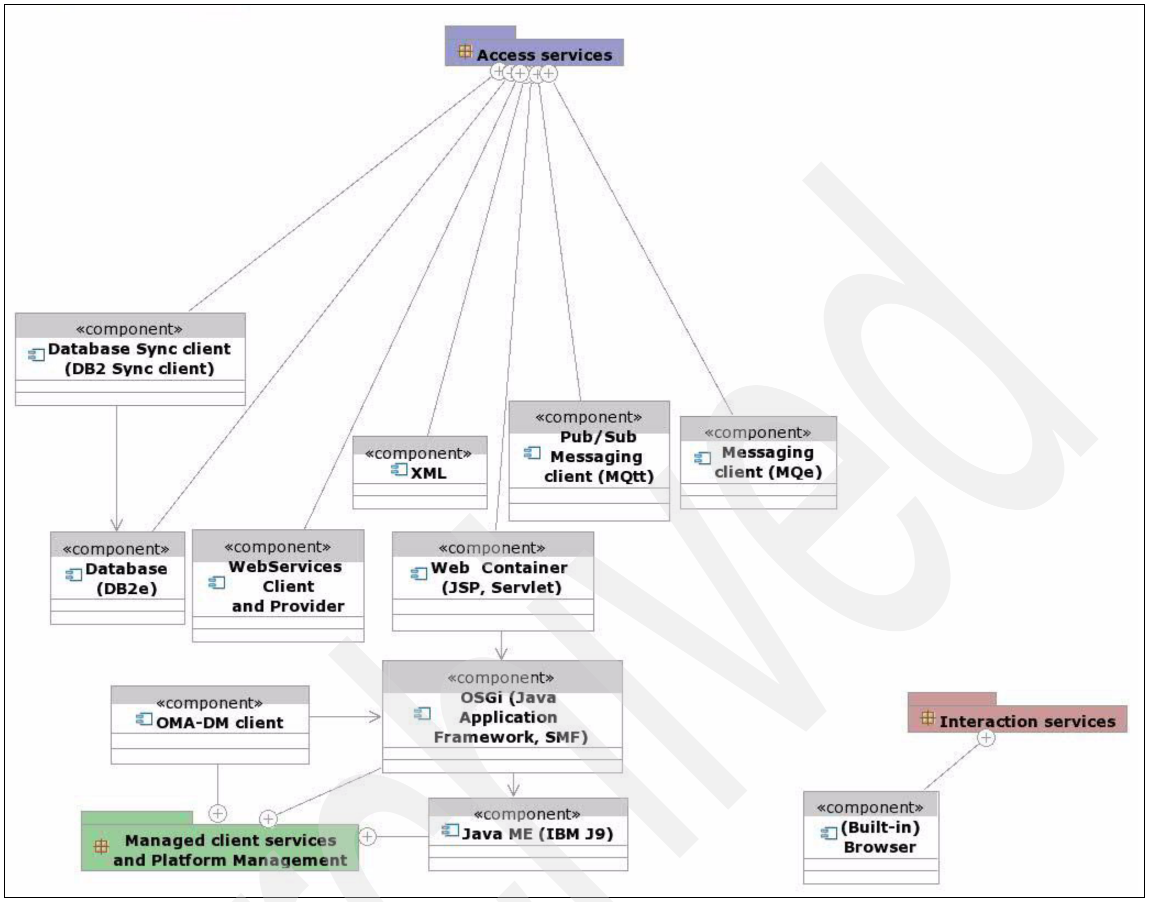

Access services
«component» Database Sync client (DB2 Sync client)
«component» XML
«component» Pub/Sub Messaging client (MQtt)
«component» Messaging client (MQe)
«component» Database (DB2e)
«component» WebServices Client and Provider
«component» Web Container (JSP, Servlet)
«component» OMA-DM client
«component» OSGi (Java Application Framework, SMF)
Interaction services
Managed client services and Platform Management
«component» Java ME (IBM J9)
«component» (Built-in) Browser

Figure 7-19 Distributed rich client services components non-PC form factor variation

7.6 Voice-Enabled Client::Runtime pattern

This section provides the details for the Runtime pattern and Product mapping for the Voice-Enabled Client application pattern.

7.6.1 Runtime pattern

The Voice-Enabled Client::Runtime pattern is used for existing or new applications that need to support voice interaction.

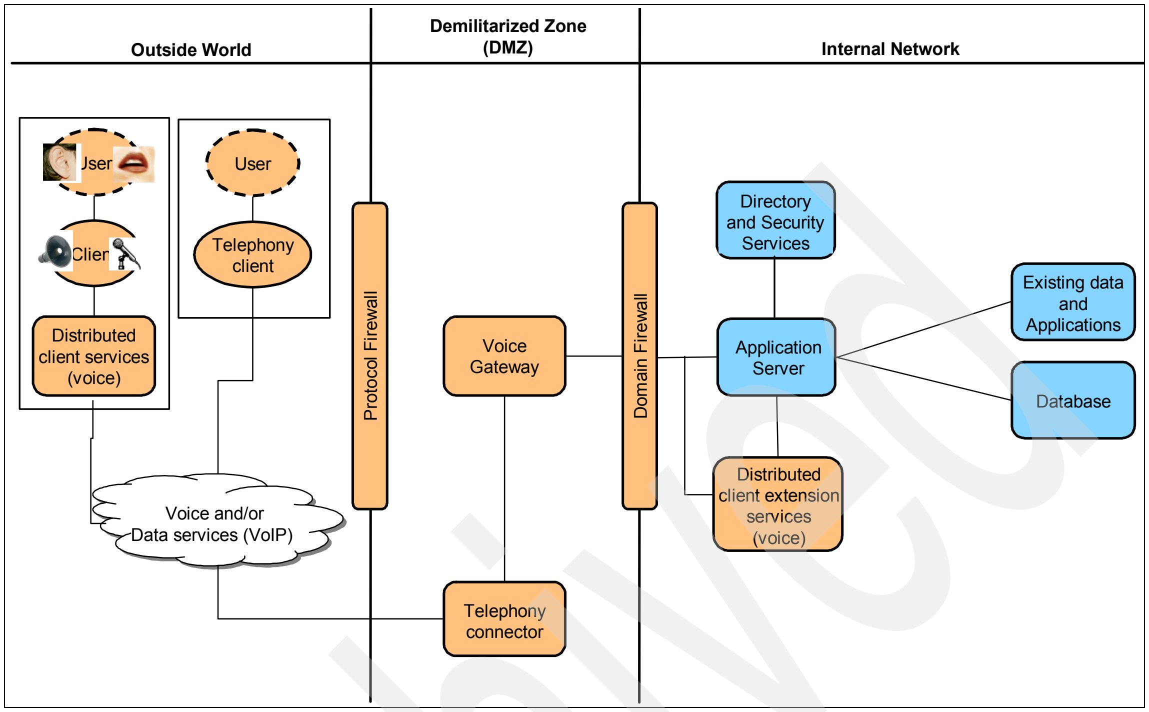

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (voice)
User
Telephony client
Voice and/or Data services (VoIP)
Protocol Firewall
Voice Gateway
Telephony connector
Domain Firewall
Directory and Security Services
Application Server
Distributed client extension services (voice)
Existing data and Applications
Database

Figure 7-20 Voice-Enabled Client::Runtime pattern for voice-enabled devices

Figure 7-20 shows the nodes that must be considered for solutions with voice interfaces. As depicted in Figure 7-20, there are two different types of clients:

- Telephony client: The telephony client is a traditional phone client. It represents any type of phone, including cell phones, landline phones, etc.
- Distributed Rich Client with VoIP[1] support: With this client voice signals are transformed into data packets and transmitted over the an IP connection to a server or another IP phone client. VoIP service providers offer the necessary server capabilities for connection setup (“switch board”) and data transmission. They provide the interfaces between the IP network and existing telephony networks so that IP phone clients can “call” traditional phone clients and vice-versa.

When looking at Figure 7-20, the key nodes that are used in voice applications are as follows:

- Voice gateway: The voice gateway node supports the call control and management via an IVR and speech technology set (speech recognition and TTS). In our current partnerships we support Genesys Enterprise Edition,

[1] VoIP - Voice over IP (also known as Internet telephony)

Avaya IR, and a range of Cisco products. We also support WebSphere Voice Response V3.1. There are other partners who are also providing IVR connectivity to IBM speech resources. It is valid to say that any VoiceXML 2.0 and MRCP client compatible system can be supported on this node.

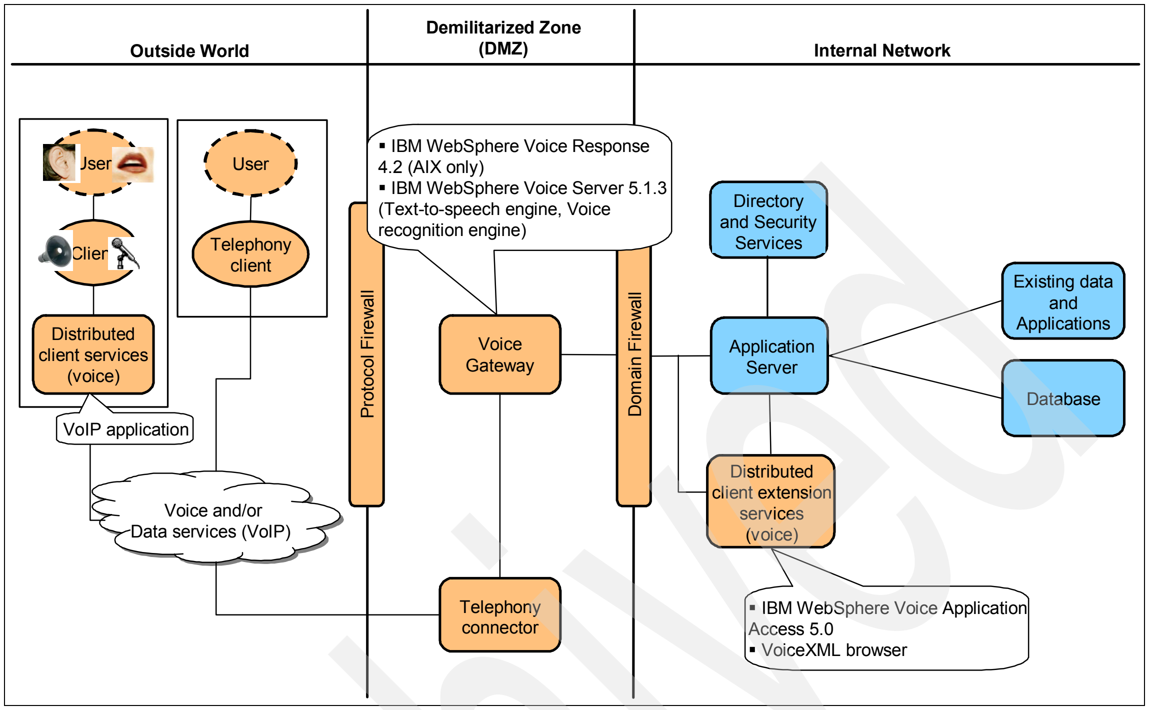
- Telephony connector: The telephony connector node is used to maintain voice-related information such as vocabularies of words, pronunciations of words, and statistics on how certain words are used. It also contains the logic for speech recognition and can convert audio into text. This node is required to connect the telephony network to enterprise telephony. The node would include any switch (PBX) and might support analogue, digital (E1/T1), and VoIP connectivity depending on enterprise requirements.
- Distributed client extension services (voice): The *Distributed client extension services (voice) node* is used to extend the Application Server node with voice capabilities for applications. This implies that the Distributed client extension services (voice) node contains components that are used for enabling voice support for the existing applications.

  Most often it means that a component can send and render the incoming and outgoing requests into the format that is suitable for voice services (for example, VoiceXML). The extension is basically an aggregator that transforms standard HTML pages and content into the VoiceXML format when the requests come through the Telephony gateway and vice-versa when responses need to be sent back through that telephony channel.

7.6.2 Product mapping

Figure 7-21 on page 178 displays the Product mapping for the Voice-Enabled Client::Runtime pattern. The key IBM software products that instantiate the voice services are as follows:

- Voice gateway:
  – IBM WebSphere Voice Response
  – IBM WebSphere Voice Server
    • Text-To-Speech engine (TTS)
    • Voice Recognition engine
- Telephony connector
- Components delivering the Distributed client extension services (voice)
  – IBM WebSphere Voice Application Access v5
  – VoiceXML Browser



Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (voice)
VoIP application
User
Telephony client
Voice and/or Data services (VoIP)
Protocol Firewall
▪ IBM WebSphere Voice Response 4.2 (AIX only)
▪ IBM WebSphere Voice Server 5.1.3 (Text-to-speech engine, Voice recognition engine)
Voice Gateway
Telephony connector
Domain Firewall
Directory and Security Services
Application Server
Existing data and Applications
Database
Distributed client extension services (voice)
▪ IBM WebSphere Voice Application Access 5.0
▪ VoiceXML browser

Figure 7-21 Voice-Enabled Client::Runtime pattern::Product mapping

The client services needed within this pattern depend on the user scenario. If a (thin) telephony client is used, no further services are required on the telephone device. If a VoIP client is used, the client platform must provide the VoIP client software, which can interact with the infrastructure of a VoIP service provider.

The difference between this Runtime pattern and the later introduced in 7.7, “Distributed Multimodal Client::Runtime pattern” on page 179, is that the Voice-Enabled Client::Runtime pattern uses the telephony infrastructure to transmit voice signals between the client and the back-end. Voice signals are translated into markup languages for application interaction by the provided Runtime nodes (see Voice gateway, telephony connector, etc.). In case of the Distributed Rich Client::Runtime pattern,=Multimodal variation, a multimodal browser on the device translates the voice signals locally on the device in VoiceXML markup language. Data transmitted between the client and the back-end is markup language based (no voice signals).

7.7 Distributed Multimodal Client::Runtime pattern

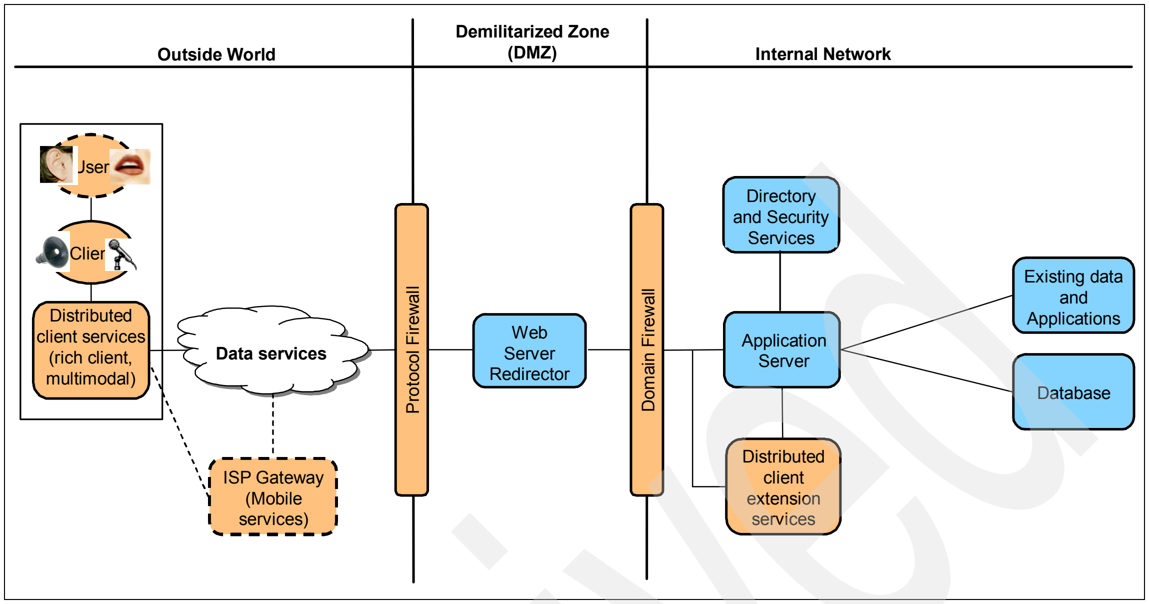
This section provides the details for the Runtime pattern, Product mapping, and component model for the Distributed Multimodal Client application pattern.

7.7.1 Runtime pattern

The Distributed Rich Client::Runtime pattern=Multimodal variation extends the Distributed Rich Client::Runtime pattern with multimodal user interaction capabilities (see Figure 7-22 on page 180). The user interface is extended with voice capabilities. Appropriate devices with a microphone and speaker are required.

The required Distributed client extension services on the server-side are the same as the Distributed Rich Client::Runtime pattern=WebUI variation. The Interaction services on the device are extended with VoiceXML capabilities (Multi-modal browser).

Note: This can also be used without local client logic; in that case, this is just an extension of the Thin Client pattern, except that the browser is voice enabled. The limitation in this usage, as with most Thin Client patterns, is that full connectivity is required. By adding the distributed client services on the client, we extend the value of multimodal to the disconnected environment.



Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (rich client, multimodal)
Data services
ISP Gateway (Mobile services)
Protocol Firewall
Web Server Redirector
Domain Firewall
Directory and Security Services
Application Server
Distributed client extension services
Existing data and Applications
Database

Figure 7-22 Distributed Rich Client::Runtime pattern=Multimodal variation

7.7.2 Product mapping

The Distributed client services and Distributed client extension services are instantiated by the following IBM products for this pattern (see Figure 7-16 on page 172):

- Server:
  - IBM WebSphere Everyplace Deployment V6.0
- Client:
  - IBM Multimodal Toolkit
  - IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux

  or

  - IBM Workplace Client Technology Micro Edition, Enterprise Offering V5.8.1 (WCTME-EO)

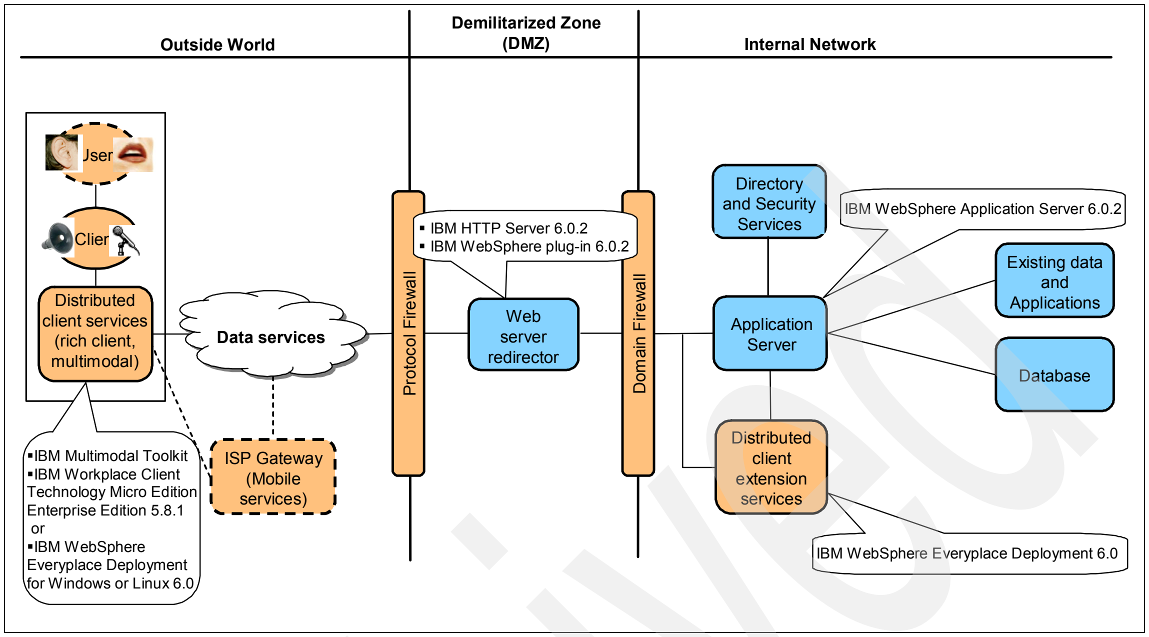

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (rich client, multimodal)
▪IBM Multimodal Toolkit
▪IBM Workplace Client Technology Micro Edition Enterprise Edition 5.8.1
or
▪IBM WebSphere Everyplace Deployment for Windows or Linux 6.0
Data services
ISP Gateway (Mobile services)
Protocol Firewall
▪ IBM HTTP Server 6.0.2
▪ IBM WebSphere plug-in 6.0.2
Web server redirector
Domain Firewall
Directory and Security Services
IBM WebSphere Application Server 6.0.2
Application Server
Existing data and Applications
Database
Distributed client extension services
IBM WebSphere Everyplace Deployment 6.0

Figure 7-23 Distributed Rich Client::Runtime pattern=Multimodal variation::Product mapping

7.7.3 Component model

The Distributed Rich Client::Runtime pattern=Multimodal variation extends the Distributed Rich Client::Runtime pattern with multimodal browser-based user interface capabilities on the client. Since the Distributed Rich Client::Runtime pattern is an extension to the Distributed Application Client::Runtime pattern, the same architectural components can be used, with the addition of the Interaction services for local user voice access on the device (multimodal browser based).

Figure 7-24 on page 183 shows the used components of the Client services from Table 6-4 on page 151. The Client services are classified in Interaction services, Access services, Managed client services, and Platform Management. Components of the Interaction services are extended with X+V browser capabilities. The following components are available on the client:

- Managed client services and Platform Management:
  - Java ME (IBM J9)
  - OSGi (IBM SMF)
  - OMA DM client (IBM Enterprise Management Agent)
- Access services:
  - Web Container (provided with WCTME-EO and WED client)
  - XML

  - Messaging client (IBM MQe)
  - Pub/Sub Messaging client (IBM MQtt)
  - Web Services Client and Provider
  - Database (IBM DB2e)
  - Database Sync client (DB2e synchronization client)
- Interaction services:
  - Multi-modal browser
  - Graphics (for the Eclipse-based WED client)

The Rich Client Application Framework standard component is based on the Eclipse standard. It contains the Interaction services, the Managed client services, and Platform Management, as well as the standardized components of the Access services. IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux implements this standard and provides additional access services (see Figure 7-24 on page 183).

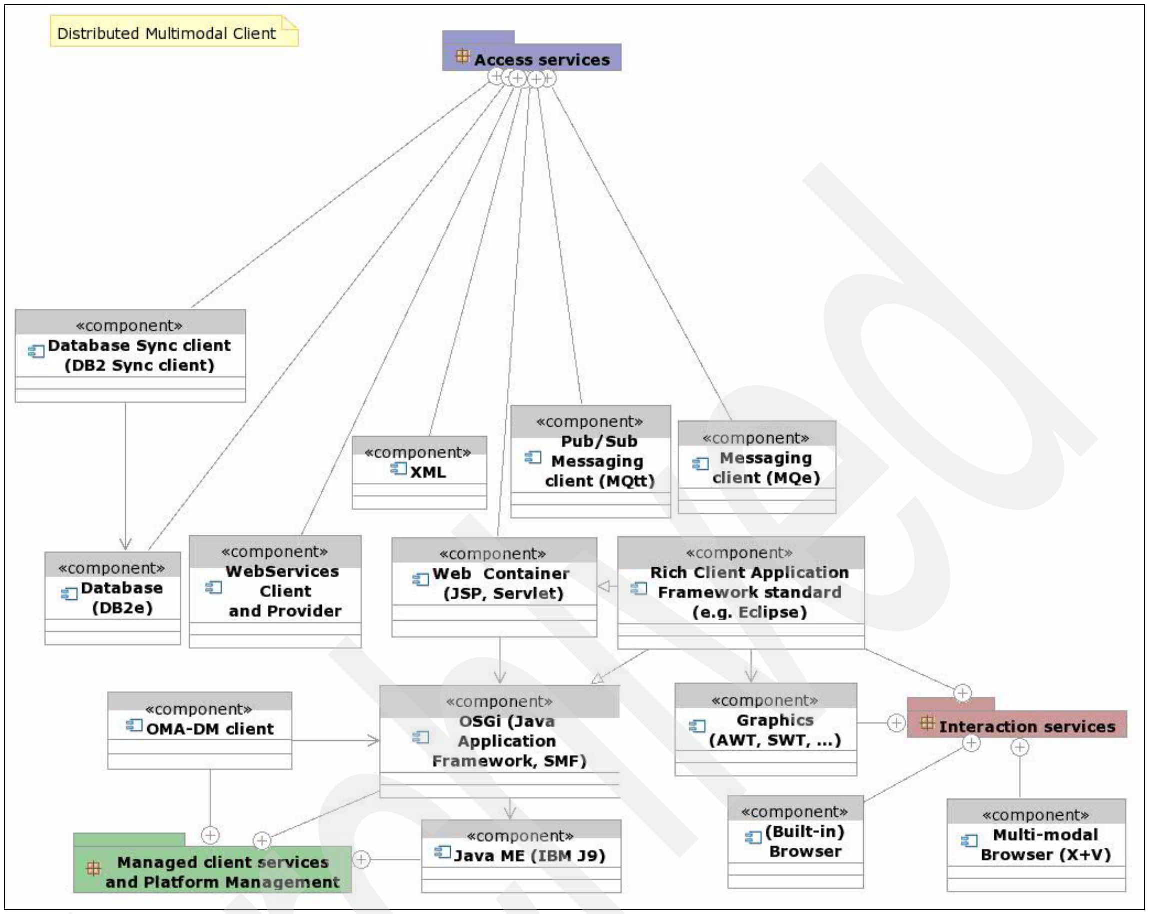

Distributed Multimodal Client
Access services
«component» Database Sync client (DB2 Sync client)
«component» XML
«component» Pub/Sub Messaging client (MQtt)
«component» Messaging client (MQe)
«component» Database (DB2e)
«component» WebServices Client and Provider
«component» Web Container (JSP, Servlet)
«component» Rich Client Application Framework standard (e.g. Eclipse)
«component» OMA-DM client
«component» OSGi (Java Application Framework, SMF)
«component» Graphics (AWT, SWT, ...)
Interaction services
«component» (Built-in) Browser
«component» Multi-modal Browser (X+V)
Managed client services and Platform Management
«component» Java ME (IBM J9)

Figure 7-24 Distributed (multimodal) client services components

7.8 Distributed Collaboration Client::Runtime pattern

This section provides the details for the Runtime pattern, Product mapping, and component model for the Distributed Collaboration Client application pattern.

7.8.1 Runtime pattern

The Distributed Collaboration Client::Runtime pattern extends the Distributed Rich Client::Runtime pattern with collaboration capabilities delivered by the Distributed collaboration services node (see Figure 7-25 on page 184).

Certain collaboration services require always connected devices such as awareness and instant messaging services. Disconnected work is possible for messaging services such as PIM and e-mail.

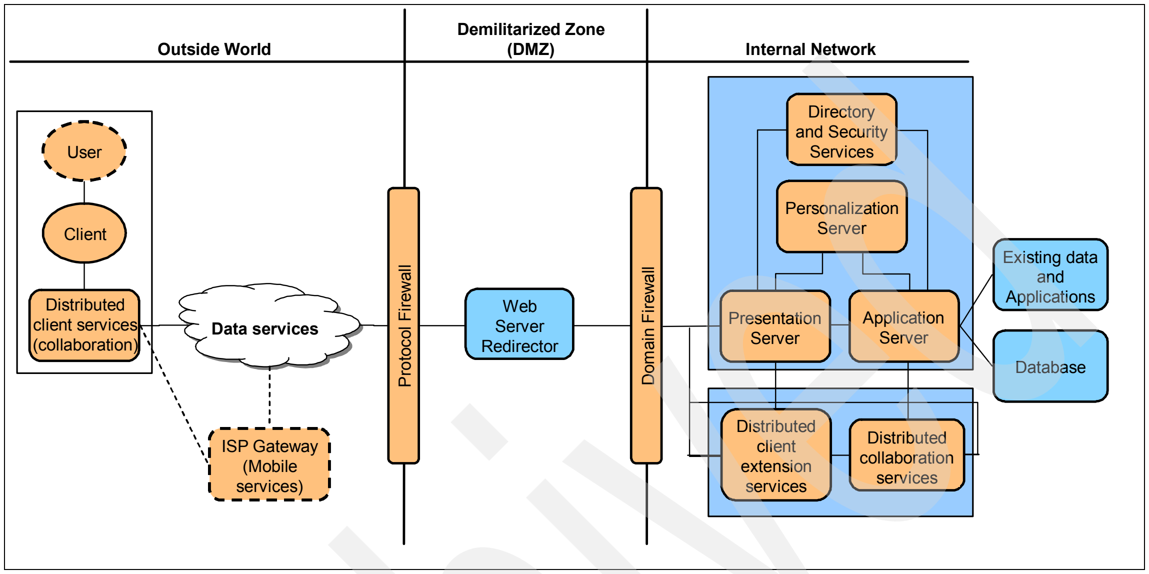

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (collaboration)
Data services
ISP Gateway (Mobile services)
Protocol Firewall
Web Server Redirector
Domain Firewall
Directory and Security Services
Personalization Server
Presentation Server
Application Server
Existing data and Applications
Database
Distributed client extension services
Distributed collaboration services

Figure 7-25 Distributed Collaboration Client::Runtime pattern

The Distributed client services provide collaboration services with rich user interface experience on the device. The Distributed client extension services on the server-side provide the following services:

- Portal services
- Content Adaptation
- Web Services
- Distributed collaboration services
  - Team collaboration services (including instant messaging and awareness)
  - Document services
  - Messaging services (including PIM/e-mail)
  - Web content management services
  - Learning services

The Distributed client services deliver the client counterparts to these services and the following additional services:

- Managed client services and Platform Management:
  - Java runtime
  - Java application framework
  - Platform Management
- Access services

  - Web Container
  - XML Parsing services
  - Notification services
  - Relational database services
- Interaction services
  - Browser
  - Graphic services

The Rich Client Application Framework standard provides standardized Client services in the Eclipse platform.

7.8.2 Product mapping

The Distributed client services and the Distributed collaboration services extended by the Distributed client extension services are instantiated by the following IBM products:

- IBM Workplace Collaboration Services V2.5.1
- IBM Workplace Managed Client

The IBM Workplace Managed Client product is based on the Eclipse standard. It extends this standard client platform with additional services to deliver team collaboration such as PIM and e-mail, instant messaging, and document management.

The IBM WebSphere Collaboration Services server provides the corresponding services to the IBM Workplace Managed Client (Client services) to ensure seamless back-end integration.

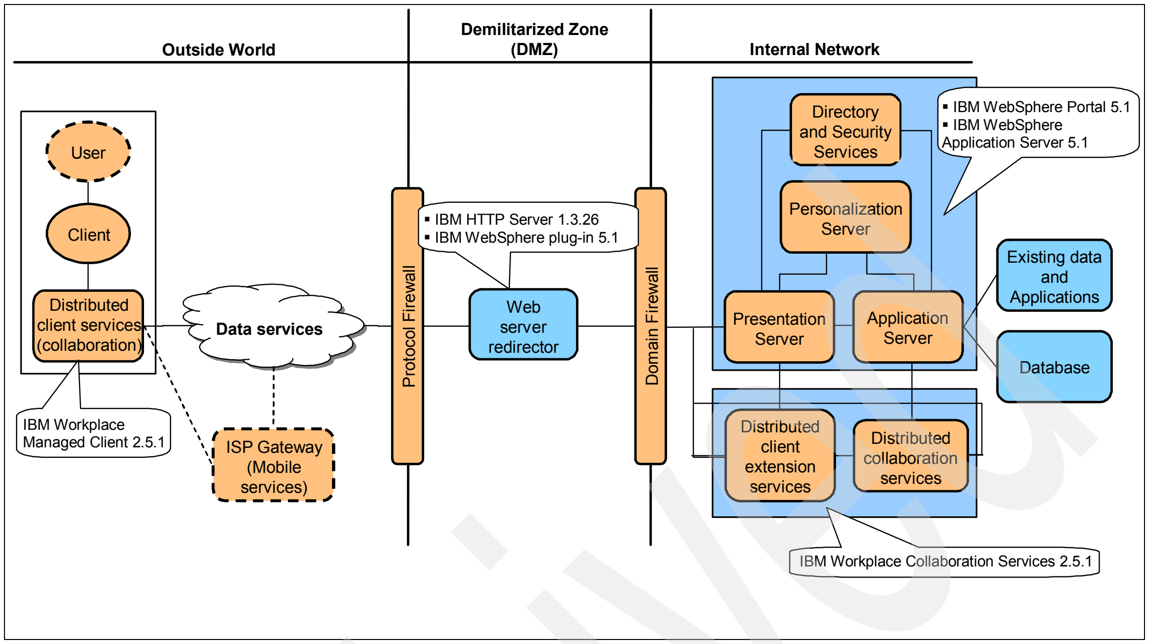

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (collaboration)
IBM Workplace Managed Client 2.5.1
Data services
ISP Gateway (Mobile services)
Protocol Firewall
IBM HTTP Server 1.3.26
IBM WebSphere plug-in 5.1
Web server redirector
Domain Firewall
Directory and Security Services
Personalization Server
Presentation Server
Application Server
IBM WebSphere Portal 5.1
IBM WebSphere Application Server 5.1
Existing data and Applications
Database
Distributed client extension services
Distributed collaboration services
IBM Workplace Collaboration Services 2.5.1

Figure 7-26 Distributed Collaboration Client::Runtime pattern::Product mapping

7.8.3 Component model

The Distributed Collaboration Client::Runtime pattern extends the Distributed Rich Client::Runtime pattern with distributed collaboration and portal services to address collaboration requirements (see Figure 7-27 on page 187). On the client-side (device), Access and Interaction services for collaboration are provided.

The high-level architectural solution components (services) are shown in Figure 7-27 on page 187. The Distributed collaboration services were included as Portal Services extensions.

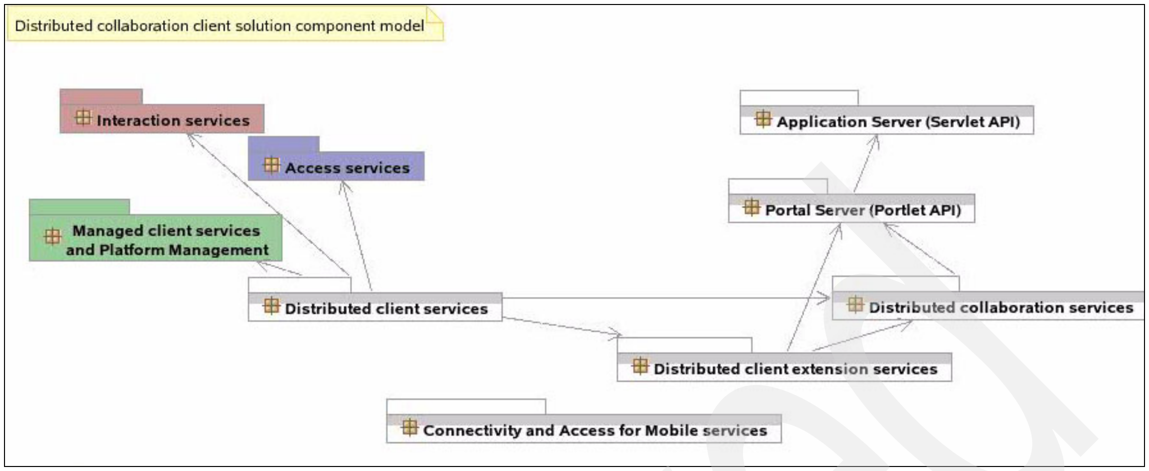

Distributed collaboration client solution component model
Interaction services
Access services
Managed client services and Platform Management
Distributed client services
Application Server (Servlet API)
Portal Server (Portlet API)
Distributed collaboration services
Distributed client extension services
Connectivity and Access for Mobile services

Figure 7-27 Architectural components of IBM Workplace Collaboration Services solution

The following components of the IBM Workplace Collaboration Services server instantiate the Distributed collaboration services and Distributed client extension services:

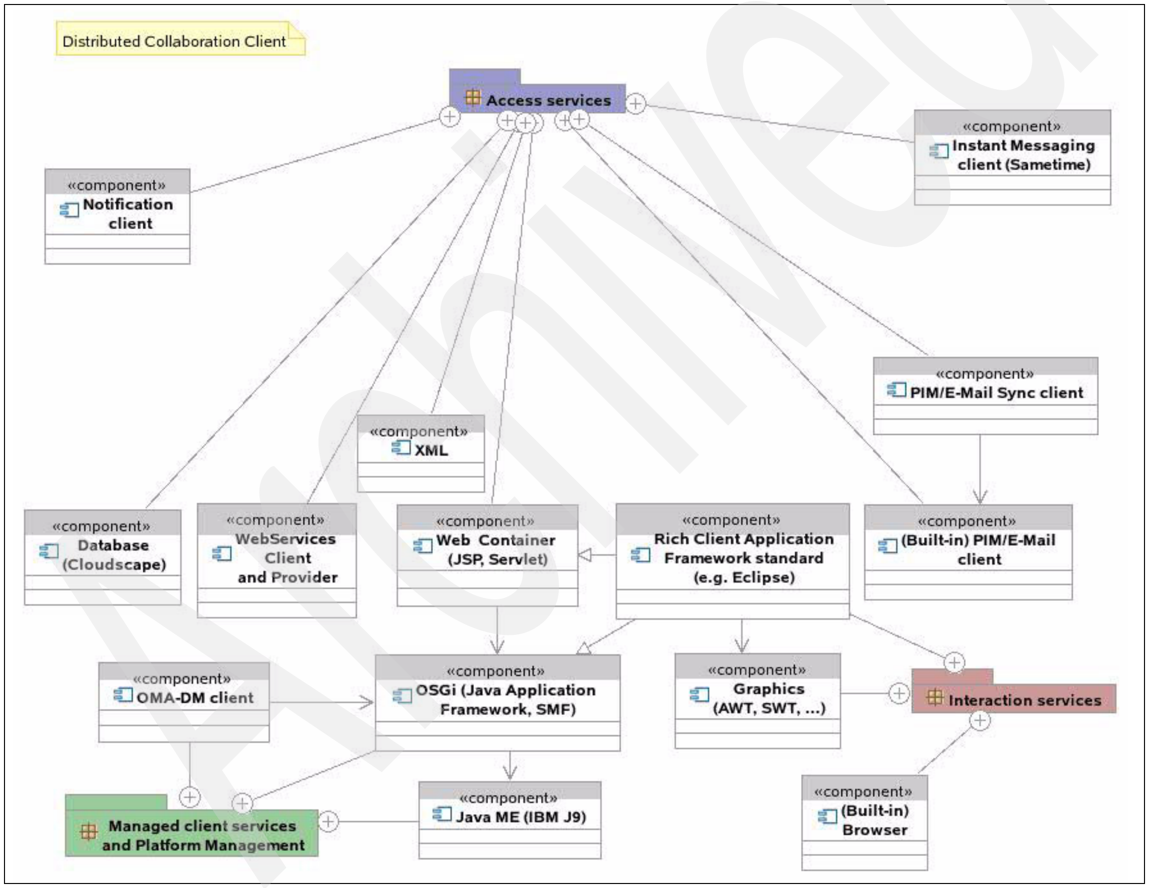
- Portal Server (extensions)
- Content Adaptation (extensions)
- Web Services Client and Provider
- Distributed collaboration services
  - Team Collaboration Services (including instant messaging)
  - Document Services
  - Messaging Services (including PIM/e-mail)
  - Web Content Management Services
  - Learning Services

Figure 7-28 on page 188 shows the used Client services components from Table 6-4 on page 151.

The Rich Client Application Framework standard component is based on the Eclipse platform. It contains the Interaction services, the Managed client services, and Platform Management, as well as the standard parts of the Access services. The collaboration services extensions for the client are provided as (Eclipse) plug-ins. Together with the IBM Eclipse implementation, they build the IBM Workplace Managed Client consisting of the following components:

- Managed client services and Platform Management:
  - Java ME (IBM J9)
  - OSGi (IBM SMF)
  - OMA DM client (IBM DM agent for OSGi and for Eclipse Plug-in)

- Access services:
  - Web Container
  - XML
  - Web Services client and provider
  - Database (IBM Cloudscape)
  - Team collaboration services Eclipse plug-ins (for example, Instant Messaging client, also usable as Notification client)
  - Messaging services Eclipse plug-ins (PIM/e-mail client, PIM/e-mail Sync client)
- Interaction services:
  - (Built-in) Browser
  - Graphics



Distributed Collaboration Client
Access services
«component» Instant Messaging client (Sametime)
«component» Notification client
«component» PIM/E-Mail Sync client
«component» XML
«component» Database (Cloudscape)
«component» WebServices Client and Provider
«component» Web Container (JSP, Servlet)
«component» Rich Client Application Framework standard (e.g. Eclipse)
«component» (Built-in) PIM/E-Mail client
«component» OMA-DM client
«component» OSGi (Java Application Framework, SMF)
«component» Graphics (AWT, SWT, ...)
Interaction services
Managed client services and Platform Management
«component» Java ME (IBM J9)
«component» (Built-in) Browser

Figure 7-28 Distributed (collaboration) client services components

7.9 Roaming Connectivity::Runtime pattern

This section provides the details for the Runtime pattern, Product mapping, and component model for the Roaming Connectivity application pattern.

7.9.1 Runtime pattern

The Roaming Connectivity::Runtime pattern covers all aspects of remote access to a secured computing environment using insecure networks and/or devices. The Roaming Connectivity::Runtime pattern provides the following secure remote connection services in addition to roaming and VPN:

- Security
  - Authentication/Authorization
  - Access control
  - Filtering (Firewall)
  - Encryption
- Session management
  - Sessions are continuously maintained although a physical connection dropout might have occurred. Applications do not break their communication in case of short network outages. In such a case, data transmission is slowed down but not lost.
  - A physical session can be terminated if no data is sent. A network connection is automatically re-established when data is pending.
  - Seamless network roaming capability.
- Optimization
  - Packet filtering
  - Compression

As seen in Figure 7-29 on page 190, the Connectivity and Access for Mobile services node is located in the DMZ. The user and configuration data for the services are stored in the central customer user directory. Client services for Connectivity and Access for Mobile services are also required for session management, optimization, and security.

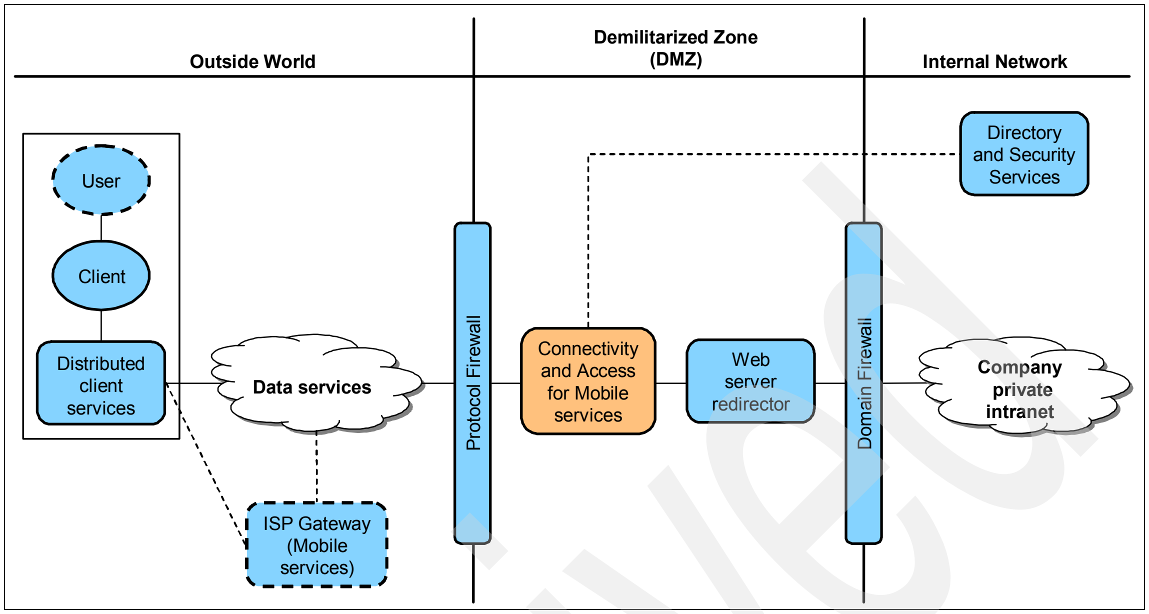
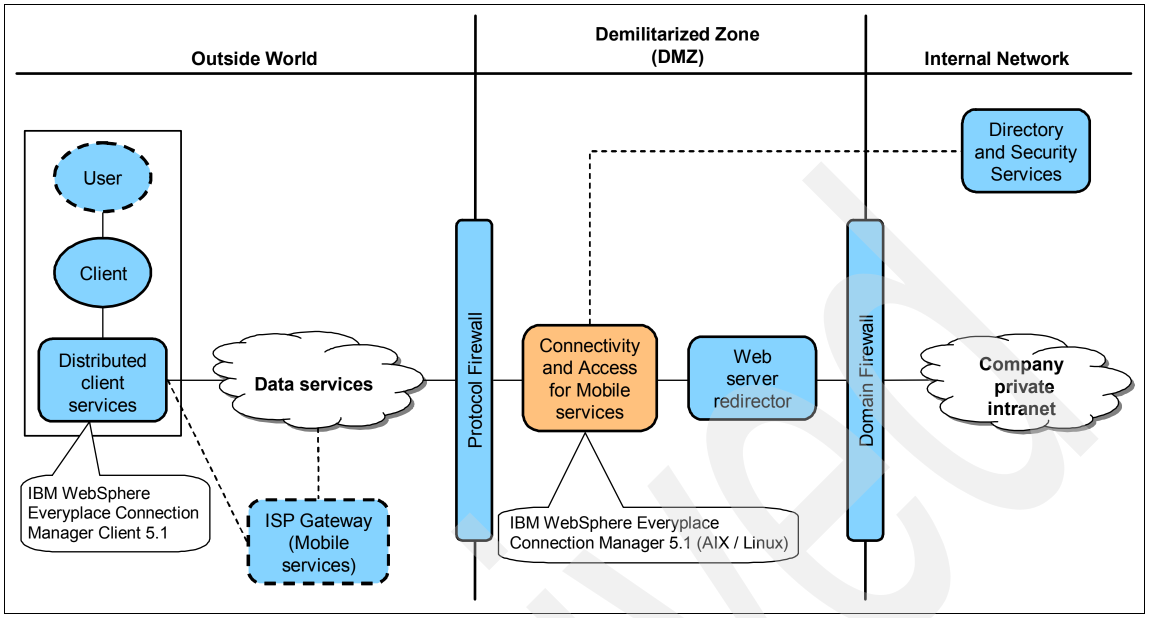


Figure 7-29 Roaming Connectivity::Runtime pattern (composable)

7.9.2 Product mapping

The Connectivity and Access for Mobile services node is instantiated by the following IBM products (see Figure 7-30 on page 191):

- IBM WebSphere Everyplace Connection Manager V5.1 (AIX or Linux)
- IBM WebSphere Everyplace Connection Manager Client V5.1



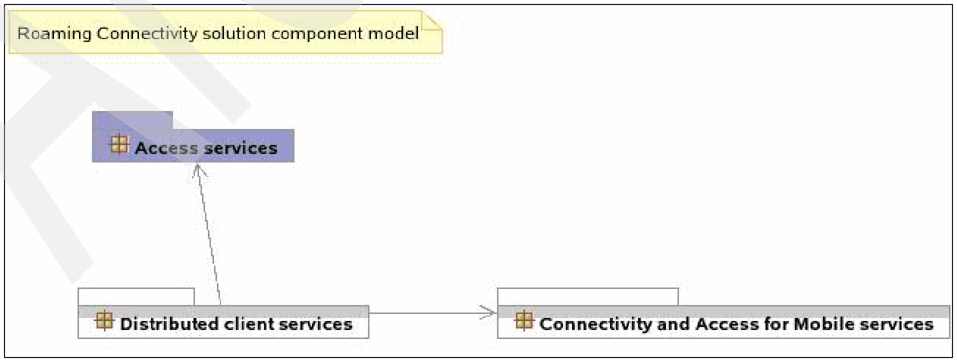
Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services
IBM WebSphere Everyplace Connection Manager Client 5.1
Data services
ISP Gateway (Mobile services)
Protocol Firewall
Connectivity and Access for Mobile services
IBM WebSphere Everyplace Connection Manager 5.1 (AIX / Linux)
Web server redirector
Domain Firewall
Directory and Security Services
Company private intranet

Figure 7-30 Roaming Connectivity::Runtime pattern::Product mapping

Component model

Figure 7-31 shows the architectural components for the Roaming Connectivity::Runtime pattern. The Distributed client services use the Connectivity and Access for Mobile services to securely and reliably access the application server infrastructure.

The client component of the Connectivity and Access for Mobile services on the device belongs to the Access services (see Figure 7-32 on page 192).



Roaming Connectivity solution component model
Access services
Distributed client services
Connectivity and Access for Mobile services

Figure 7-31 Architecture components for Roaming Connectivity

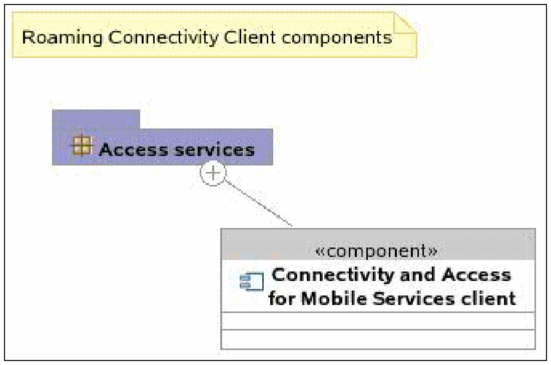

Roaming Connectivity Client components
Access services
«component»
Connectivity and Access for Mobile Services client

Figure 7-32 Distributed (roaming connectivity) client services components

7.10 Composite::Runtime pattern

This section provides the details for the Runtime pattern, Product mapping, and component model for a Composite application pattern.

7.10.1 Runtime pattern

The Composite::Runtime pattern comprises nodes from different Runtime patterns to fulfill user requirements that cannot be satisfied with a single Runtime pattern.

The majority of the user requirements for a rich user interface lead to the Distributed Rich Client application pattern and the corresponding Distributed Rich Client::Runtime pattern. For example, a customer requests instant messaging services in addition to the rich client requirements. The instant messaging functions are delivered by the Distributed collaboration services with its Distributed client extension services and nodes (see 7.8, “Distributed Collaboration Client::Runtime pattern” on page 183). If the instant messaging service included in the Distributed collaboration services is added to the Distributed Rich Client::Runtime pattern, a Composite runtime pattern is needed (see Figure 7-33 on page 193).

Another typical candidate for a Composite::Runtime pattern is the Roaming Connectivity::Runtime pattern. The majority of access solutions require secure remote access functions (for example, VPN functions, session management, and traffic optimization). The nodes from the Roaming Connectivity::Runtime pattern are simply added to the Runtime pattern that satisfies the other (not connection-related) customer requirements.

The Runtime patterns out of which the Composite pattern were created deliver their Distributed client services and their Distributed client extension services to build the nodes of the Composite::Runtime pattern.

In the example displayed in Figure 7-33, the Distributed client extension services gather the functions delivered by the extension services for the Distributed Rich Client::Runtime pattern and the Distributed Collaboration Client::Runtime pattern:

- Distributed Rich Client::Runtime pattern services
  - Relational database services (includes database synchronization)
  - Assured messaging services
  - Device services
- Distributed Collaboration Client::Runtime pattern services
  - Portal services
  - Content Adaptation
  - Web Services
  - Distributed collaboration services
    - Team collaboration services (includes instant messaging, awareness)
    - Document services
    - Messaging services (includes PIM, e-mail)
    - Web content management services
    - Learning services

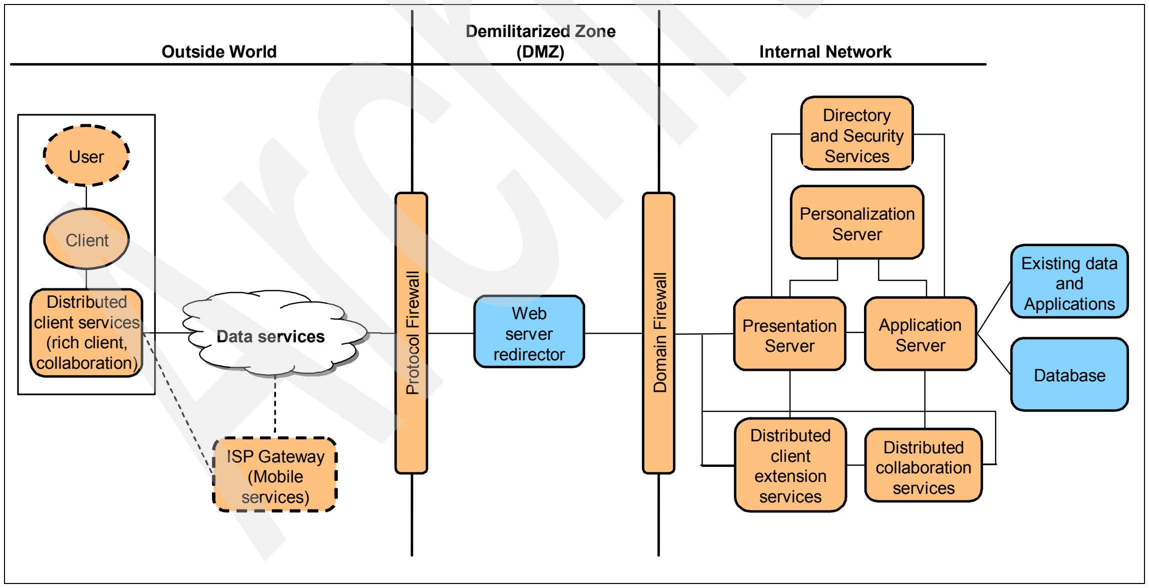

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (rich client, collaboration)
Data services
ISP Gateway (Mobile services)
Protocol Firewall
Web server redirector
Domain Firewall
Directory and Security Services
Personalization Server
Presentation Server
Application Server
Distributed client extension services
Distributed collaboration services
Existing data and Applications
Database

Figure 7-33 Composite::Runtime pattern (Distributed Rich client and Distributed Collaboration Client)

The Distributed client services contain the following client counterparts of the above-listed services and additional services:

- Common services (delivered by Composition of Client services):
  - Managed client services and Platform Management
    - Java runtime
    - Java application framework
    - Platform Management
  - Access services
    - Web Container
    - XML Parsing services
    - Web Services
    - Relational database services
  - Interaction services
    - Browser
    - Graphic services
- Distributed Rich Client::Runtime pattern
  - Access services
    - Assured messaging services
    - Pub/Sub messaging services
- Distributed Collaboration Client::Runtime pattern
  - Access services
    - Team collaboration services (for example, instant messaging)
    - Messaging services (PIM/e-mail)

The Rich Client Application Framework standard component is based on the Eclipse platform standard and realizes the Interaction services, the Managed client services, and Platform Management, as well as the standardized components of the Access services.

7.10.2 Product mapping

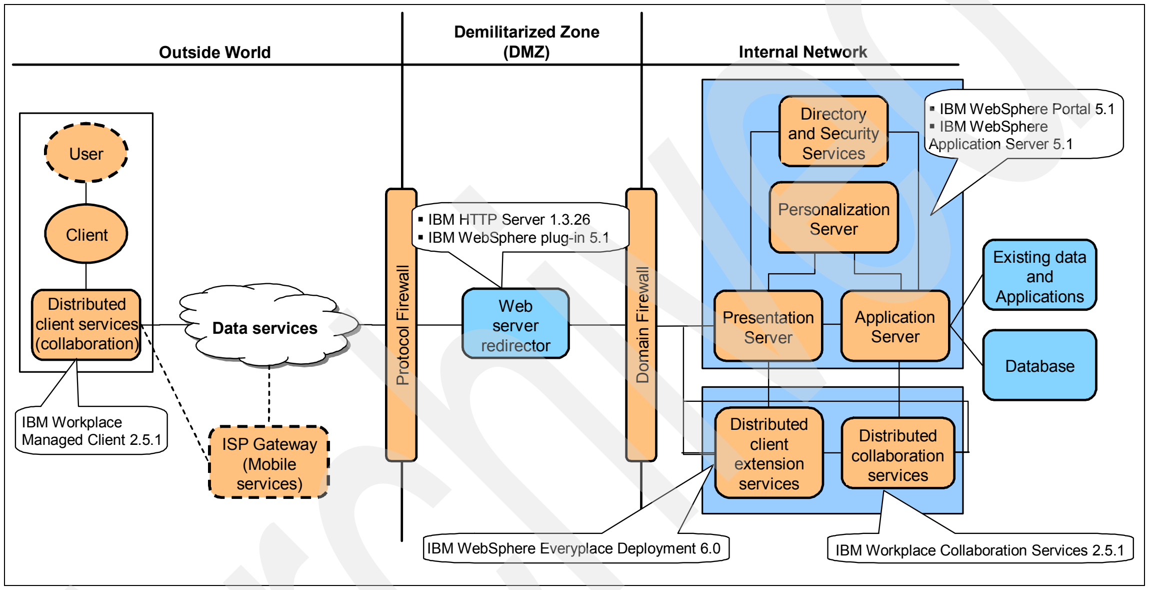
The Distributed client extension services in the Composite::Runtime pattern are instantiated by the following IBM products (see Figure 7-34 on page 195):

- IBM WebSphere Everyplace Deployment V6.0
- IBM Workplace Collaboration Services V2.5.1
- IBM Workplace Managed Client V2.5.1

These Distributed client extension services enable existing and new applications to be accessed with different (mobile) devices. They also further extend the Distributed collaboration services with mobile access capabilities. The Distributed collaboration services are delivered by the IBM Workplace Collaboration Services server.

On the client (device) side, the Distributed client services are provided by IBM Workplace Managed Client software. This product is based on the Eclipse standard. It contains additional services (plug-ins) to enable collaboration functions on the device (for example, PIM/e-mail and instant messaging). This also enables the common synchronous programming model for the server and the client.

The IBM Workplace Collaboration Services provide the corresponding services to the Client services on the server-side to ensure seamless back-end integration.

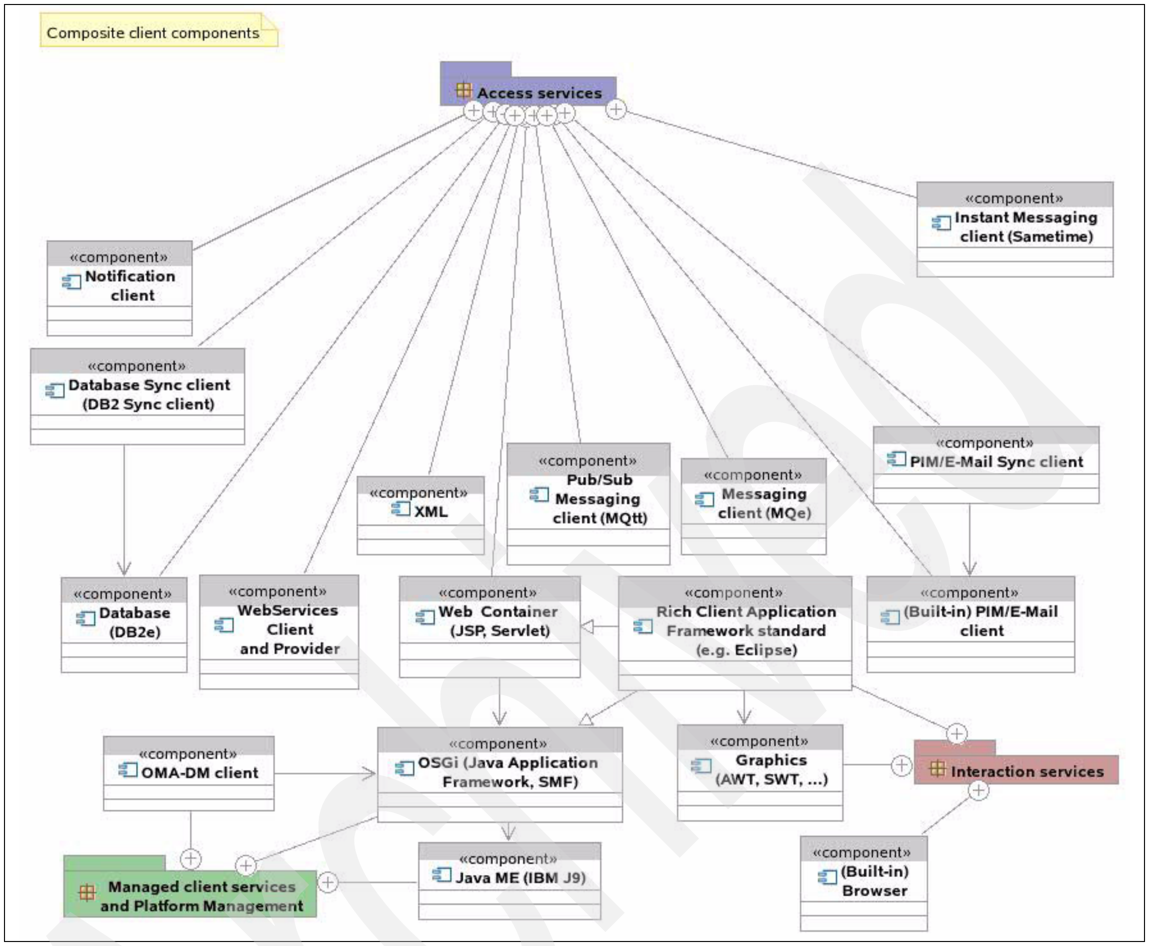


Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (collaboration)
IBM Workplace Managed Client 2.5.1
Data services
ISP Gateway (Mobile services)
Protocol Firewall
IBM HTTP Server 1.3.26
IBM WebSphere plug-in 5.1
Web server redirector
Domain Firewall
Directory and Security Services
Personalization Server
Presentation Server
Application Server
IBM WebSphere Portal 5.1
IBM WebSphere Application Server 5.1
Existing data and Applications
Database
Distributed client extension services
Distributed collaboration services
IBM WebSphere Everyplace Deployment 6.0
IBM Workplace Collaboration Services 2.5.1

Figure 7-34 Composite::Runtime pattern::Product mapping

7.10.3 Component model

Figure 7-35 on page 196 displays the component model for the Composite::Runtime pattern.



Composite client components
Access services
«component» Notification client
«component» Database Sync client (DB2 Sync client)
«component» Instant Messaging client (Sametime)
«component» XML
«component» Pub/Sub Messaging client (MQtt)
«component» Messaging client (MQe)
«component» PIM/E-Mail Sync client
«component» Database (DB2e)
«component» WebServices Client and Provider
«component» Web Container (JSP, Servlet)
«component» Rich Client Application Framework standard (e.g. Eclipse)
«component» (Built-in) PIM/E-Mail client
«component» OMA-DM client
«component» OSGi (Java Application Framework, SMF)
«component» Graphics (AWT, SWT, ...)
Interaction services
Managed client services and Platform Management
«component» Java ME (IBM J9)
«component» (Built-in) Browser

Figure 7-35 Distributed client services components (composition out of Rich and Collaboration client)

8 Requirements analysis and solution design

A key objective of the ITSO working example is to demonstrate how the reusable patterns assets are used within RUP to accelerate the solution design.

The business scenario presented in this chapter is for the fictitious ITSO Insurance company. The scenario highlights how the Distributed Rich Client application pattern is used to extend the existing SOA enterprise application to the edge. The solution provides claims adjusters with the capability to work on-site with the customer in a disconnectable mode, and sync data with the enterprise when connected to the network.

This chapter presents the ITSO Insurance company initial context and functional requirements. We then analyze these requirements to produce use cases identifying the user interactions with the system. Next, we demonstrate how to use the IT requirements as criteria to select the appropriate Application patterns, Runtime patterns, and Product mapping.

This chapter is organized into the following sections:

- Business modeling
- IT requirements
- Use cases
- Apply patterns to the solution design

8.1 Business modeling

The business modeling includes a description of the initial context of the ITSO Insurance company system, and a definition of the business requirements for a new mobile capable system.

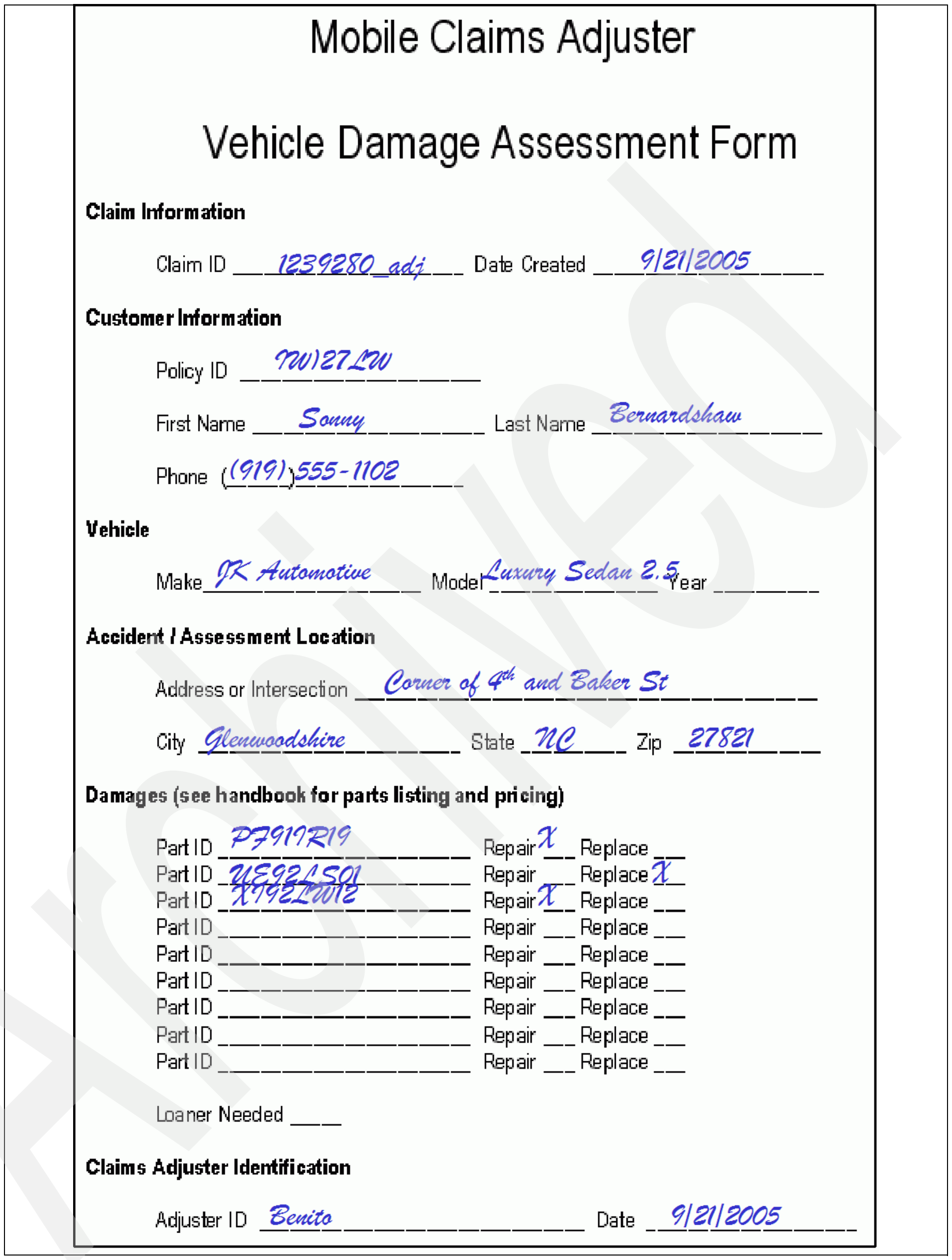
8.1.1 Initial context

The ITSO Insurance company currently uses a Web application accessible to thin clients (browser based) to process auto insurance claims.

When a customer gets into an auto accident he calls the insurance agent to report the accident with the objective of opening an insurance claim. The customer policy is retrieved from the online Web application and then the agent creates a new auto insurance claim in the system.

An adjuster is then dispatched to the scene of the accident to manually capture the claim information in a paper form and take a photo of the vehicle damage. This process entails the adjuster looking up parts information from a paper catalog (Vehicle Repair Handbook) to determine replacement cost of parts, and recording the part numbers on the paper form (see Figure 8-1 on page 199). In some cases a part code is not found in the paper Vehicle Repair Handbook (because it is out of date), and the adjuster then needs to call the data center service desk to retrieve the requested part number from the database.

Once the claims adjuster is back in the office, she submits paper claims forms captured for the day to the data center to be keyed into the online system. This process is time consuming and labor intensive. In addition, the data quality is subject to error due to the nature of transferring handwritten information from a form and re-keying this into an electronic system. This step is often the most critical in the whole process because any posted error would affect payment. In addition, there is no a cross-check method in place. In the case of a suspected error, the adjuster has to be called back for clarification.



Mobile Claims Adjuster

Vehicle Damage Assessment Form

Claim Information

Claim ID 1239280_adj Date Created 9/21/2005

Customer Information

Policy ID 7W)27LW

First Name Sonny Last Name Bernardshaw

Phone (919) 555-1102

Vehicle

Make JK Automotive Model Luxury Sedan 2.5 Year

Accident / Assessment Location

Address or Intersection Corner of 4th and Baker St

City Glenwoodshire State NC Zip 27821

Damages (see handbook for parts listing and pricing)

Part ID	PF911R19	Repair X	Replace
Part ID	UE92LS01	Repair	Replace X
Part ID	X192LW12	Repair X	Replace
Part ID		Repair	Replace
Part ID		Repair	Replace
Part ID		Repair	Replace
Part ID		Repair	Replace
Part ID		Repair	Replace
Part ID		Repair	Replace

Loaner Needed

Claims Adjuster Identification

Adjuster ID Benito Date 9/21/2005

Figure 8-1 Current paper auto claim form captured by the Adjuster

8.1.2 Business challenges of existing system

This section outlines the key business challenges of the existing system that need to be considered as problems to address in a new system.

- Time processing the claim
  - The current paper-based forms process does not support a responsive dialog with the customer.
  - The end-to-end process requires at least a few days to be realized.
- Flexibility and currency of vehicle catalog data
  - The parts catalog data published in printed form in the Vehicle Repair Handbook is not flexible. The catalog is not well structured for parts information retrieval, and updates to parts information require publishing a new paper catalog, which is expensive and untimely.
- Claims data accuracy
  - Recording the parts data from the Vehicle Repair Handbook to the Damage Assessment form is hand written on a paper form, and later manually keyed into the online system. This process is prone to error.
  - Discrete systems require repetitive data entry.
- Customer satisfaction
  - The ITSO Insurance company is not able to provide an immediate response to the customer's auto claim.
  - The claim process can take several days since the data needs to be manually keyed into the system.
  - The current paper and manual process does not easily facilitate making the customer aware of process status changes.
  - The adjuster is currently not able to calculate damage value, since the pricing information is not part of Vehicle Repair Handbook (only part numbers).

8.1.3 Business requirements

The business requirements in this section are modeled on the new claims business process desired by the ITSO as a result of business process optimization.

New claims business process

The ITSO has developed a new claim business process that requires extending the current back-end system with mobile capabilities.

A key requirement for customer satisfaction is the ability to process low-dollar amount claims (for example, $500 or less) the same day the claim is submitted.

The adjusters currently have laptops with wireless connectivity and the ITSO would like to leverage the capabilities of this device. With the new process, the adjuster will capture the claims data electronically (via laptop) on-site with the customer and eliminate the manual paper forms data capture process. Figure 8-2 on page 202 outlines the key steps in the new ITSO auto insurance claims process.

The basic process flow of the new claims system is described as follows (see Figure 8-2 on page 202):

1. Customer calls Agent to report accident and open claim.
2. Agent retrieves customer profile and opens a claim in the online system. Claim is assigned to an available adjuster.
3. Adjuster dispatched to customer site to electronically capture claims data and digital photo.
4. Adjuster electronically synchronizes claims data with back-end system when connected to a network.
5. Claim available in online system for agent to report to customer the status of the claim.
6. Claim complete.

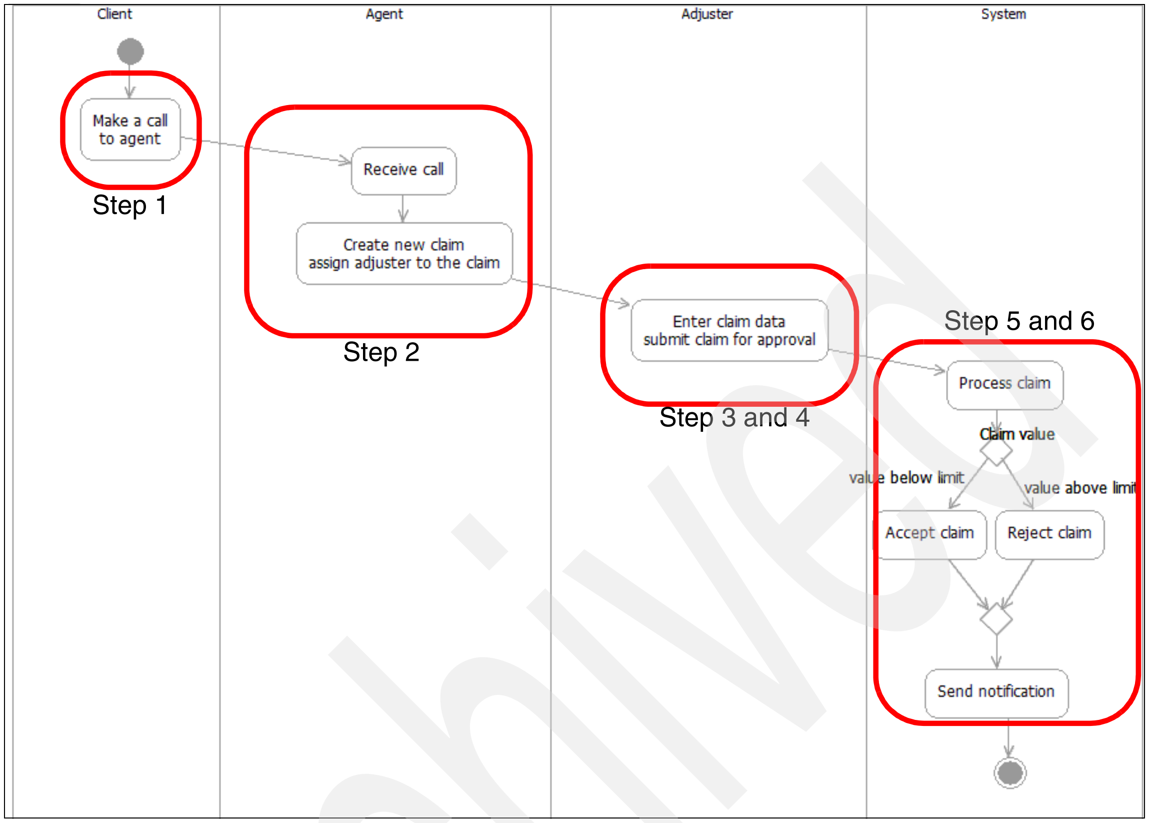


Figure 8-2 Basic Automated Claim Business Process flow

Business requirements

The proposed solution needs to conform to the new business process and provide for a higher level of process automation.

The ITSO Insurance business requirements are as follows:

1. The new extended business process needs to allow adjusters to enter claims data electronically to facilitate prompt processing of claim.
2. Eliminate error-prone paper form and data re-keying process.
3. Provide the auto parts catalog (Vehicle Repair Handbook) in electronic form to the adjuster to allow for more accessible parts information and data that is always up-to-date.
4. The new business process should extend existing back-end and online Web applications. In addition to leveraging existing systems, this allows for a more seamless model for the ITSO IT organization to create the new system.

5. Improve client satisfaction by providing immediate feedback and progress notification.

8.2 IT requirements

This section describes the functional and non-functional requirements for the ITSO Insurance scenario.

8.2.1 Functional requirements

In the previous section we captured the business requirements. We now have to further refine the customer requirements to the level of technical IT requirements.

The functional requirements for the ITSO scenario are as follows:

- The existing enterprise application needs to be extended to allow the adjuster to capture claims data electronically. The adjuster will use their laptop and local application to capture the customer claim information on-site. The agent will continue to use the existing online Web application.
- The system has to provide the capability for the adjuster to work remotely in disconnected mode using the necessary data stored locally on their device.
- The requirement is to provide all the features listed below for mobile users:
  - User has to be able to create and fill in a new claim record on-site with the customer. Created claims have to be submitted to the back-end system whenever they are completed and connection to a network is established.
  - The adjuster must be able to attach a vehicle damage picture (digital photo) to the claim when she enters the claim data. The picture will be stored on the adjuster's device separate from the claim. Pictures need to be uploaded to the enterprise upon request (for example, when the mobile device is connected to enterprise network).
  - The mobile user desires to be notified about status changes of the claims while he is working on-site (provided connectivity available).
  - The disconnectable application on the mobile device needs to generate a message to be sent to the enterprise system when a new claim is submitted for approval.
  - All data stored on the mobile device has to be protected by local encryption.
  - All data communication between the disconnectable client application and back-end systems has to be protected against interception.

8.2.2 Non-functional requirements

This section presents the non-functional requirements for the ITSO Insurance scenario.

Security

Due to the fact that the system is processing user-sensitive data, it is necessary to pay close attention to the security aspects of the system. The system must preserve stored data from misuse or discovery by an unauthorized person. Central user authentication is required to grant access to any level of the system.

The system must further protect confidentiality and integrity of data transferred between a user's device and back-end system and avoid potential modifications by a third party.

Device management

The ITSO has several device management related requirements:

- Remote installation of application code on the user's workstation is required. New versions of the application, fixes, and configuration changes have to be distributed to the user's laptop remotely.
- Inventory management capability is required.
- The administrator should have tools to manage and schedule administrative tasks.

8.3 Use cases

This section further defines the business and IT requirements in the form of use cases. We first define the actors of the system and then provide a summary of the use cases relevant to our scenario. In addition, we have documented each use case in table form, and provided a corresponding activity diagram.

8.3.1 Actors

Based on the requirements defined, we have identified the following actors:

- Customer
- Agent
- Adjuster
- System
- Administrator

Customer in this case is the auto insurance policy holder. The customer will interact with the insurance agent. In our scenario, the customer will contact the agent to report accidents and open an insurance claim.

The *agent* is an internal user of the ITSO Insurance company. The agent works out of a regional office and is primarily responsible for interacting with the insurance policy holder (customer). Most notably for our scenario, the agent responds to calls from the customer to open new auto insurance claims in the event of an accident.

Adjuster is an internal user of the ITSO Insurance company. An adjuster is assigned a claim, and is dispatched to the customer site to analyze the damages to the vehicle. The adjuster will use his laptop with the disconnectable client application to capture the claim information electronically, and then synchronize the data with the back-end system when connected.

System represents the services provided by the enterprise application and middleware services.

Administrator is the person responsible for performing the administrative tasks for the ITSO Insurance system. Although we have provided a summary of the use cases for the administrator, we do not provide detailed use cases.

8.3.2 Use case model

The use case model seen in Figure 8-3 on page 206 shows the functions of the application and the relationship to the actors. In our example, we have identified two human actors (agents, adjusters) who interact with the system.

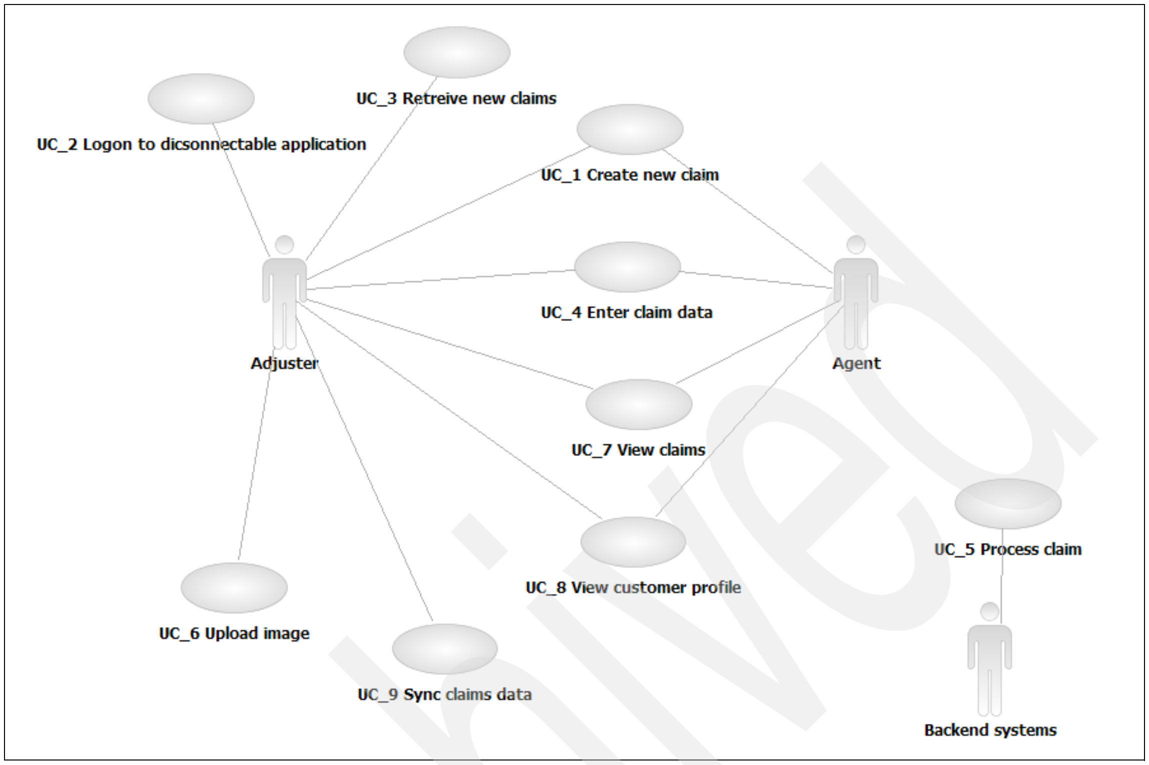


Figure 8-3 Use case model (business)

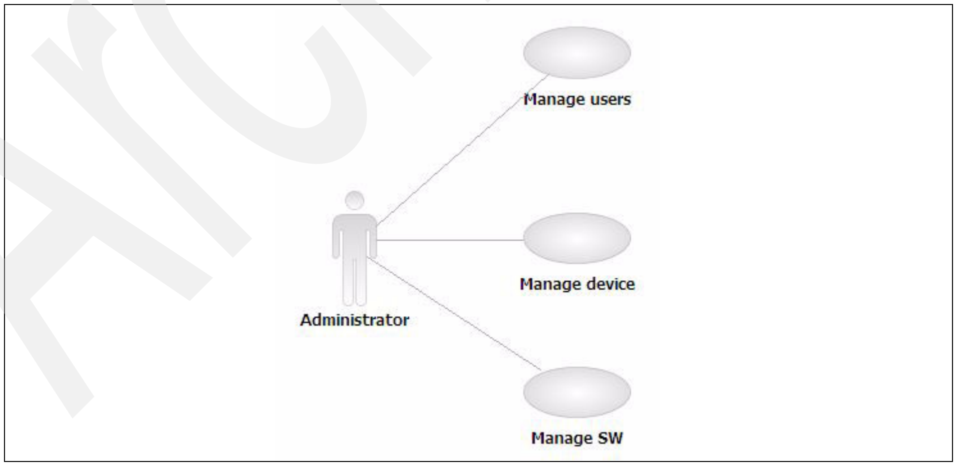


Figure 8-4 Use case model (administrative)

The use cases describe the functionality requirements of the system. They are used by the architect to design the system. Figure 8-1 provides a summary of the use cases for the ITSO Insurance scenario.

Table 8-1 Use case summary

Use case ID	Use case name	Goal in context	Primary actor	Secondary actor
UC_1	Create new claim	Agent receives call from customer and creates new claim form in online Web application. Alternatively, the adjuster can create a claim in the disconnectable application.	Agent	Adjuster
UC_2	Logon to disconnectable client application	The adjuster logs on to the disconnectable client application.	Adjuster	
UC_3	Retrieve new claims	The adjuster performs data refresh from disconnectable client application to retrieve new claims from server.	Adjuster	
UC_4	Enter claim data	The adjuster enters claim information electronically in disconnectable client application on-site with customer. Alternatively, the agent or adjuster can enter claim information in the online application.	Adjuster	Agent
UC_5	Process claim	System receives claim information and processes claim according to business rules.	System	
UC_6	Upload image	The adjuster uploads digital image of accident damages to back-end system.	Adjuster	
UC_7	View claims	The agent can view claims in the online system to report claims status to customer. Alternatively, the adjuster can view claims from the disconnectable client application.	Agent	Adjuster
UC_8	View customer profile	The agent can view a customer's profile in the online application. Alternatively, the adjuster can view the customer profiles (if claim exists for the customer) in the disconnectable application.	Agent	Adjuster
UC_9	Sync claims data	The adjuster synchronizes the claim data with the database on the device with database on server.	Adjuster	

Use case ID	Use case name	Goal in context	Primary actor	Secondary actor
UC_10	Refresh claim status	The adjuster initiates retrieval and display of all messages of claim status on the client device.	Adjuster	

The following sections detail each use case in table form and provide a corresponding activity diagram.

UC_1: Create new claim

Table 8-2 UC_1: Create new claim

Subject area	Business
Description	► A customer calls the insurance agent to report an accident. The agent retrieves the customer profile and creates a new claim. ► Alternatively, the adjuster can create a claim in the disconnectable application (that is, multiple car pile up, new claims reported on scene).
Primary actor	Agent
Secondary actor	Adjuster
Preconditions	► Customer has contacted the insurance agent to report an accident with the intention of opening a claim. ► User is successfully logged in to the application.
Termination outcome	A new claim entered into the system for approval.
Main use case scenario	1. Agent retrieves customer profile from online application. 2. Agent opens new claim in online application. 3. Agent assigns new claim to adjuster.
Alternative scenario	1. Customer contacts adjuster while on the scene of an accident (that is, multiple car pile up). 2. Adjuster retrieves customer profile from disconnectable client application. 3. Adjuster creates new claim in the disconnectable client application.

Figure 8-5 on page 209 displays an activity diagram for the create new claim use case.

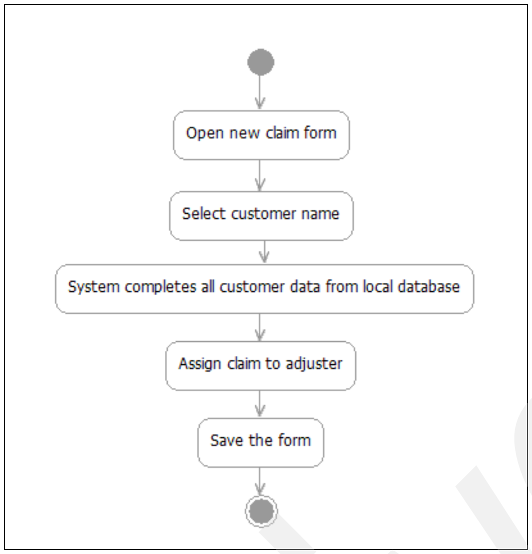


Figure 8-5 Create new claim activity diagram

UC_2: Log on to disconnectable client application

Table 8-3 UC_2: Log on to disconnectable client application

Subject area	Business
Description	Adjuster logs on to disconnectable client application
Primary actor	Adjuster
Secondary actor	
Preconditions	
Termination outcome	Adjuster is logged in successfully and credentials stored for data sync purposes in another use case
Main use case scenario	1. User enters their user ID and password 2. User clicks Login button
Alternative scenario	

Figure 8-6 on page 210 displays an activity diagram for the logon use case.

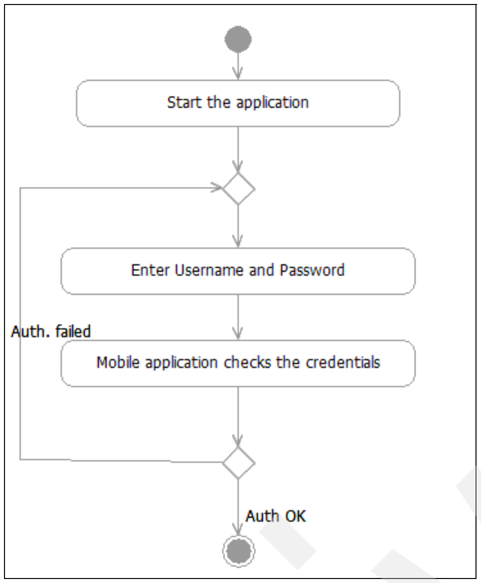

Start the application
Enter Username and Password
Auth. failed
Mobile application checks the credentials
Auth OK

Figure 8-6 Logon activity diagram

UC_3: Retrieve new claims

Table 8-4 UC_3: Retrieve new claims

Subject area	Business
Description	Adjuster retrieves new claims from the enterprise system to the disconnectable client application
Primary actor	Adjuster
Secondary actor	
Preconditions	► New claim exists to retrieve (see “UC_1: Create new claim” on page 208) ► Adjuster has successfully performed a logon (see “UC_2: Log on to disconnectable client application” on page 209)
Termination outcome	New claims are retrieved (synchronized from server to client)
Main use case scenario	1. User clicks Refresh Data button of disconnectable client application 2. Synchronization engine performs data sync (in this case server to client) 3. Informative message is displayed in disconnectable client application

Subject area	Business
Alternative scenario	

Figure 8-7 displays an activity diagram for the retrieve new claims use case.

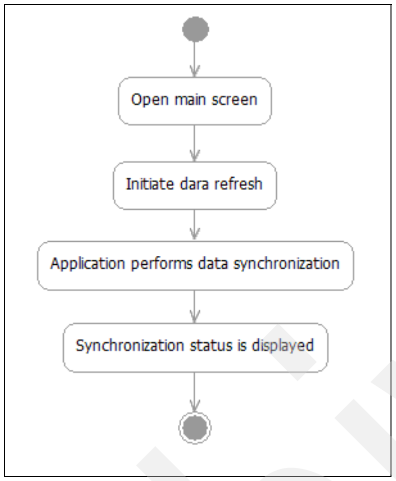

Open main screen
Initiate dara refresh
Application performs data synchronization
Synchronization status is displayed

Figure 8-7 Retrieve new claims activity diagram

UC_4: Enter claim data

Table 8-5 UC_4: Enter claim data

Subject area	Business
Description	► User enters claim information electronically in disconnectable client application on-site with customer. ► Alternatively, the agent or adjuster can enter claim information in the online application.
Primary actor	Adjuster.
Secondary actor	Agent.
Preconditions	► New claim exists to retrieve (see “UC_1: Create new claim” on page 208). ► Adjuster has successfully performed a logon (see “UC_2: Log on to disconnectable client application” on page 209). ► Claim has been retrieved (see “UC_3: Retrieve new claims” on page 210).

Subject area	Business
Termination outcome	Customer claim information is entered electronically in disconnectable client application. A message is automatically sent to initiate the approval process. Note that the message will be transferred when the disconnectable application is connected to a network (wireless or wired).
Main use case scenario	1. User opens prepared claim in disconnectable client application. 2. Click the Compute damage button. 3. User enters claim information (selects damages for defined parts; optionally adds descriptive text). 4. User optionally adds a digital photo of the vehicle to the claim. 5. User clicks submit to confirm the claim data. The claim data is stored locally in the relational database. It will be synchronized to the server in a later step. **Note:** The image (digital photo) is uploaded to the back-end in a separate process (see “UC_6: Upload image” on page 214).
Alternative scenario	1. User opens prepared claim in online application. 2. Click the Compute damage button. 3. User enters claim information (selects damages for defined parts; optionally adds descriptive text). 4. User submits completed claim for approval. **Note:** The online application does not provide the ability to upload an image.

Figure 8-8 on page 213 displays an activity diagram for the enter claim information use case.

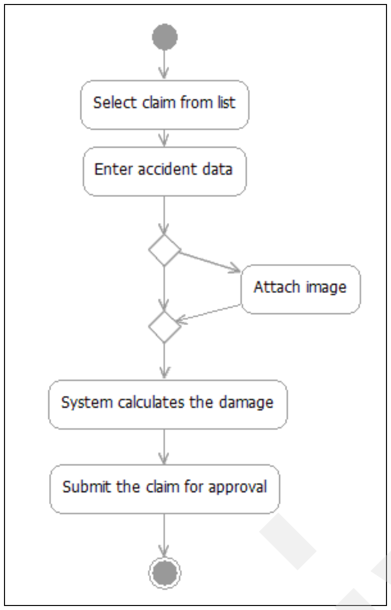


Figure 8-8 Enter claim data activity diagram

UC_5: Process claim

Table 8-6 UC_5: Process claim

Subject area	Business
Description	System receives claim information and processes claim according to business rules.
Primary actor	System.
Secondary actor	
Preconditions	► Claim has been submitted (see "UC_4: Enter claim data" on page 211).
Termination outcome	Claim approved or rejected.
Main use case scenario	1. Claim is received. 2. Process the claim according to claim value. 3. Approve or reject claim. 4. Send claim approval/rejection message.
Alternative scenario	

Figure 8-8 displays an activity diagram for the process claim use case.

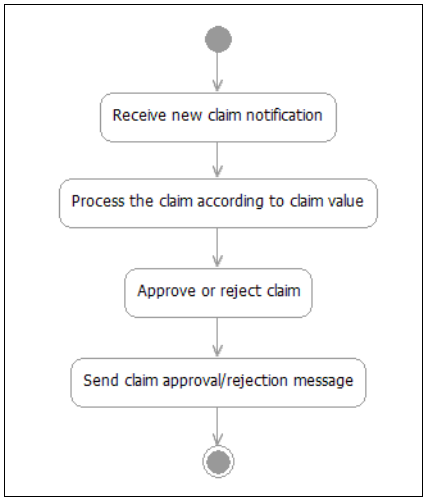

Receive new claim notification
Process the claim according to claim value
Approve or reject claim
Send claim approval/rejection message

Figure 8-9 Process claim activity diagram

UC_6: Upload image

User uploads digital image of accident damages to back-end system.

Table 8-7 UC_6: Upload image

Subject area	Business
Description	User uploads digital image of accident damages to back-end system.
Primary actor	Adjuster.
Secondary actor	
Preconditions	Image to be uploaded exists in the disconnectable client application (see "UC_4: Enter claim data" on page 211).
Termination outcome	Image is successfully uploaded to the back-end system.
Main use case scenario	1. User opens the disconnectable images application. 2. Select the appropriate Upload image button beside the image you wants to upload to the back-end server.
Alternative scenario	

Figure 8-10 on page 215 displays an activity diagram for the upload image use case.

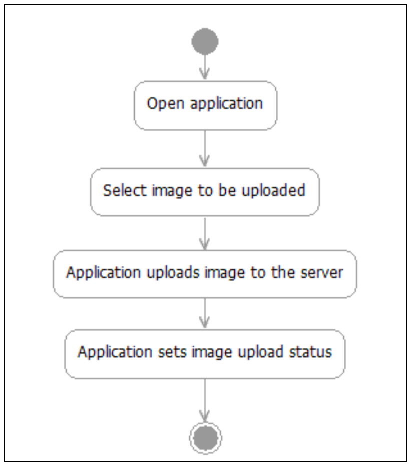

Open application
Select image to be uploaded
Application uploads image to the server
Application sets image upload status

Figure 8-10 Upload image activity diagram

UC_7: View claims

Table 8-8 UC_7: View claims

Subject area	Business
Description	► User wants to view claims in the online system to report claims status to customer. ► Alternatively, the adjuster can view claims from the disconnectable client application.
Primary actor	Agent.
Secondary actor	Adjuster.
Preconditions	Claim exists in back-end system.
Termination outcome	The claim can be retrieved.
Main use case scenario	1. Click the View claim within the online application. 2. Report status of claim to customer.
Alternative scenario	1. Refresh data. 2. Click the View claim within the disconnectable client application. 3. Report status of claim to customer.

Figure 8-11 on page 216 displays an activity diagram for the view claims use case.

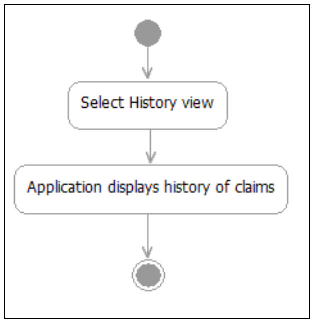


Figure 8-11 View claim activity diagram

UC_8: View customer profile

Table 8-9 UC_8: View customer profile

Subject area	Business
Description	► User wants to view a customer's profile in the online application. ► Alternatively, the Adjuster can view the customer profiles (if claim exists for the customer) in the disconnectable application.
Primary actor	Agent.
Secondary actor	Adjuster.
Preconditions	User exists as a customer in the system.
Termination outcome	The user profile is retrieved and displayed.
Main use case scenario	1. Click View Customers in the online application. 2. Select the desired customer to display the customer profile and claim information (claim information synchronized in "UC_9: Sync claims data" on page 217).
Alternative scenario	1. Click View Customers in the online application. 2. Select the desired customer to display the customer profile and claim information.

Figure 8-12 on page 217 displays an activity diagram for the view customer profile use case.

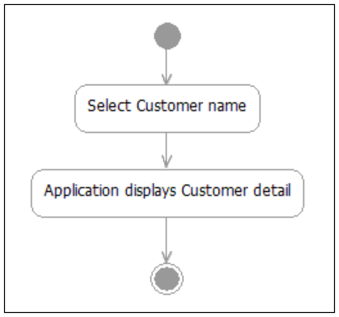


Figure 8-12 View customer profile activity diagram

UC_9: Sync claims data

Table 8-10 UC_9: Sync claims data

Subject area	**Business**
Description	User synchronizes the claim data with the database on the device with database on server.
Primary actor	Adjuster.
Secondary actor	
Preconditions	Claim exists in local database to sync (see "UC_4: Enter claim data" on page 211).
Termination outcome	New claims are synchronized with the enterprise.
Main use case scenario	1. User clicks Refresh Data button of disconnectable client application. 2. Synchronization engine performs data sync (in this case client to server). 3. Informative message is displayed in disconnectable client application.
Alternative scenario	

Figure 8-7 on page 211 displays an activity diagram for the sync claims data use case.

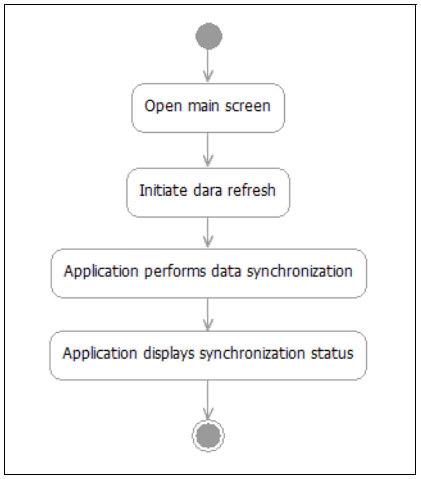

Open main screen
Initiate dara refresh
Application performs data synchronization
Application displays synchronization status

Figure 8-13 Sync claims data activity diagram

UC_10: Refresh claim status

Table 8-11 UC_10: Refresh claim status

Subject area	Business
Description	User initiates retrieval and display of all messages of claim status on the client device.
Primary actor	Adjuster.
Secondary actor	
Preconditions	
Termination outcome	All messages on device are displayed.
Main use case scenario	1. User clicks Check Message Queue. 2. Application will retrieve all messages from local store. 3. Application displays all retrieved messages.
Alternative scenario	

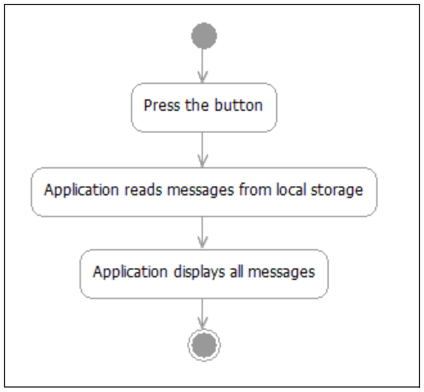

Press the button
Application reads messages from local storage
Application displays all messages

Figure 8-14 Refresh messages activity diagram

8.4 Apply patterns to the solution design

One of the key objectives of this chapter is to demonstrate how to apply the Access Integration patterns within the RUP methodology. Figure 8-15 on page 220 highlights the reusable assets and process for using the patterns to create the solution design. In the following sections we navigate this process given the defined requirements for the ITSO Insurance scenario.

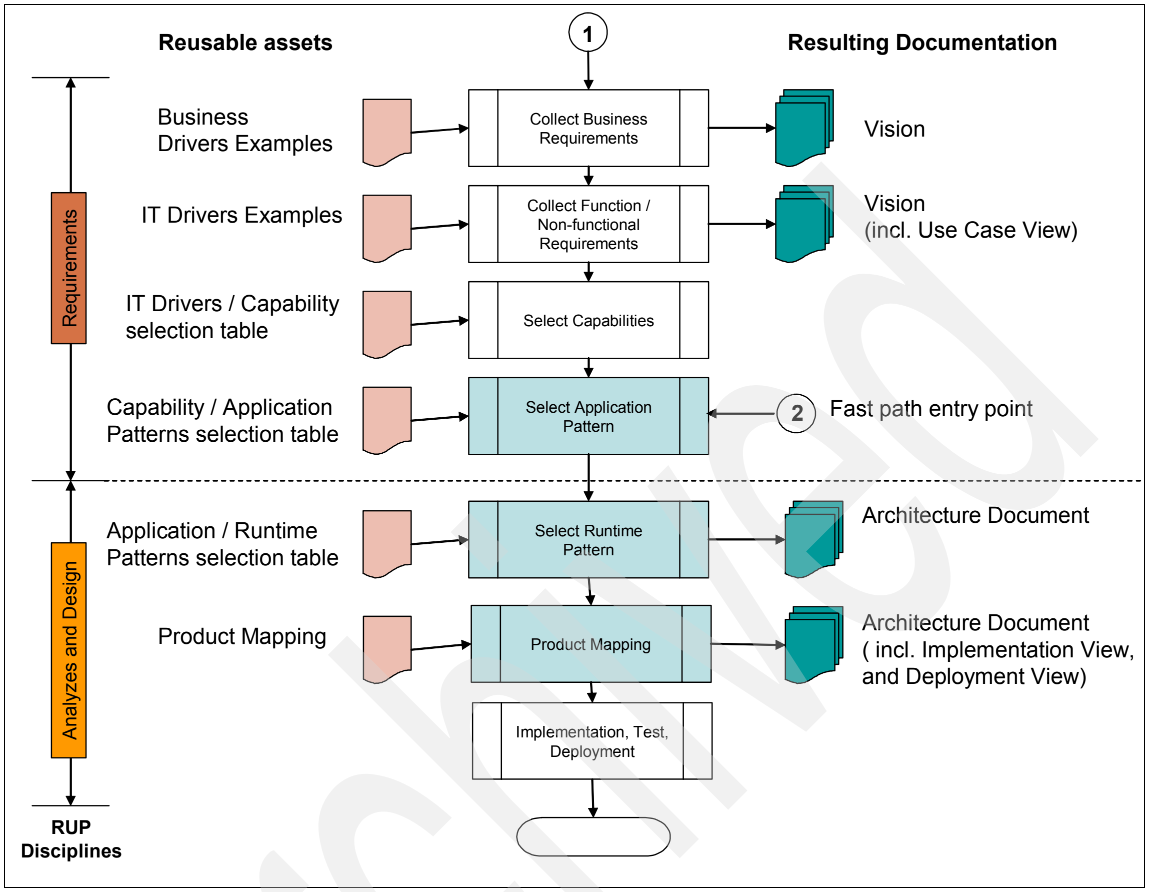

Reusable assets
1
Resulting Documentation
Business Drivers Examples
Collect Business Requirements
Vision
IT Drivers Examples
Collect Function / Non-functional Requirements
Vision (incl. Use Case View)
Requirements
IT Drivers / Capability selection table
Select Capabilities
Capability / Application Patterns selection table
Select Application Pattern
2
Fast path entry point
Application / Runtime Patterns selection table
Select Runtime Pattern
Architecture Document
Analyzes and Design
Product Mapping
Product Mapping
Architecture Document (incl. Implementation View, and Deployment View)
Implementation, Test, Deployment
RUP Disciplines

Figure 8-15 Process for using the patterns within RUP

8.4.1 Select capabilities based on requirements

In our working example, we have captured customer requirements in “IT requirements” on page 203 and modeled the requirements in “Use cases” on page 204. We are now ready to use these requirements as input criteria to select the appropriate capabilities, which ultimately map to the Application pattern or patterns.

Map customer requirements to IT drivers

Table 8-12 on page 221 maps the ITSO customer requirements to IT drivers we defined in 4.3, “IT drivers” on page 87. If you are less familiar with the Access Integration pattern, this step is helpful in providing guidance on selecting IT drivers.

Table 8-12 Mapping of customer requirements to IT drivers

Customer requirements	**IT Drivers**
Speed up development and implementation process.	**IT03** Common programming model.
Alert users of application state changes.	**IT09** Alert user/application of defined state changes.
Provide online and off-line access to application and data.	**IT19** Disconnectable application with Web User Interface.
Ability to create new database record (new claim) in disconnected mode.	**IT20** Disconnectable relational database sync services.
Ability to update existing database record (claim processing) in disconnected mode.	**IT20** Disconnectable relational database sync services.
Synchronize database on mobile device with back-end system.	**IT20** Disconnectable relational database sync services.
Attach external file to the database record (that is, picture) and upload this file to back-end system.	**IT22** Client Web services.
Notify back-end system about transaction in a disconnected mode.	**IT21** Extend the messaging paradigm by enabling transactional services.
Device user authentication based on provided user name and password.	
Protection of data integrity.	**IT20** Disconnectable relational database sync services.
Protection of data privacy.	**IT20** Disconnectable relational database sync services.
Protection of data authenticity.	**IT20** Disconnectable relational database sync services.
Remote application installation.	**IT15** Remote maintenance of applications.
Remote application maintenance.	**IT15** Remote maintenance of applications.
Remote device inventory checking.	**IT13** Inventory.
Remote middleware maintenance.	**IT14** Remote maintenance of middleware services.
Application response time and performance.	**IT32** Disconnectable app with performance focus.

Some IT drivers fulfill more than one customer requirement (see Table 8-12 on page 221). For consolidation purposes, we have provided a summary list of defined IT drivers representing the ITSO Insurance scenario requirements:

- IT03 Common programming model
- IT09 Alert user/application of defined state changes
- IT13 Inventory
- IT14 Remote maintenance of middleware services
- IT15 Remote maintenance of applications
- IT19 Disconnectable application with Web User Interface
- IT20 Disconnectable relational database sync services
- IT21 Disconnectable transactional messaging services
- IT22 Client Web services
- IT32 Disconnectable app with performance focus

Select capabilities based on IT drivers

In this step we select the capabilities based on the scenario IT drivers defined in “Map customer requirements to IT drivers” on page 220. This process is described in greater detail in 4.5, “Select capabilities based on IT drivers” on page 95. The capabilities are used as a helper to get from the IT requirements to more technically succinct capabilities, which ultimately are used to select the appropriate Application pattern or patterns.

Table 8-13 on page 223 table represents the relevant IT drivers from Table 4-3 on page 95, and the mapping to the defined capabilities.

Table 8-13 Selected capabilities given ITSO customer IT driver requirements

	Capability																						
	Multi device - Simple browsing	**Multi device - Role-based browsing**	**Location Aware Service**	**Voice User Interface**	**Online multi-modal access**	**Receive event notification**	**Subscribe to event notification**	**Receive/process actionable alerts**	**Remote mgmt, of configurations**	**Remote mgmt of middleware**	**Remote mgmt of applications**	**Disc. Web content presentation**	**Disc. forms processing**	**Disc. headless application**	**Disc. app. local WEB UI**	**Disc. app. relational sync**	**Disc app, trxl messaging**	**Disc app with Web services**	**Disc. app. local rich UI**	**Disc. app. multimodal UI**	**IM & People awareness**	**Disc. PIM and e-mail access**	**Distributed collaboration**
IT driver	**C01**	**C02**	**C03**	**C04**	**C05**	**C06**	**C07**	**C08**	**C09**	**C10**	**C11**	**C12**	**C13**	**C14**	**C15**	**C16**	**C17**	**C18**	**C19**	**C20**	**C21**	**C22**	**C23**
Alert user/app of state change																	X						
Inventory									X	X	X												
Remote maint of middleware on the device										X													
Remote maint app on device											X												
Disconnectable app w/ Web UI															X								
Disconnectable database on device w/ sync																X							
Disconnectable transactional messaging																	X						
Client Web services																		X					
Disconnectable app with focus on performance															X	X	X						

We marked the relevant capabilities in Table 8-13 on page 223 by shading the cells. The following summary lists the key capabilities that apply to the ITSO Insurance scenario:

- C09 Remote management of configurations
- C10 Remote management of middleware
- C11 Remote management of applications
- C15 Disconnectable application Web UI
- C16 Disconnectable application relational sync
- C17 Disconnectable application transactional messaging
- C18 Disconnectable application Web services

8.4.2 Select the Application pattern

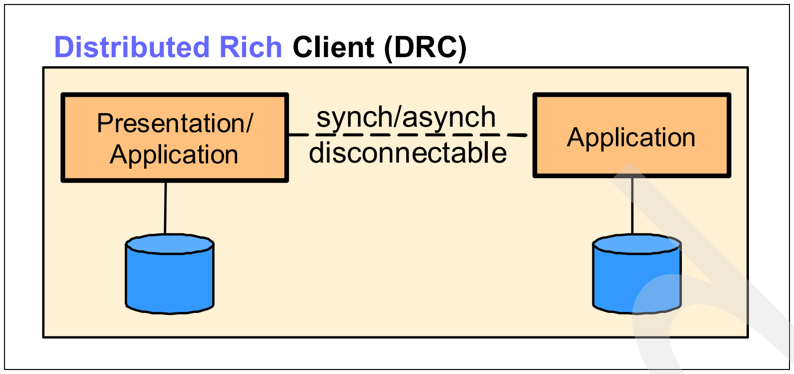
Now that we have established the capabilities needed to fulfill the customer requirements, we are ready to select the appropriate Application pattern or patterns. A description of selecting the Application pattern based on the capabilities criteria can be found in 5.2, “Select the Application pattern” on page 103.

To make this process easier to understand and see visually for our example scenario, we only included the relevant capabilities in Table 5-2 on page 105.

Table 8-14 Select the Application pattern for the given capabilities

	Capabilities	**Application patterns**						
	X - capability provided by the Application pattern (X) - capability depends on device and connectivity features	**Thin Client**	**Voice Enabled Client**	**Distributed Presentation Client**	**Distributed Application Client**	**Distributed Rich Client**	**Distributed Collaboration Client**	**Distributed Multi-modal Client**
C07	Subscription based event notifications	(X)	(X)	(X)	(X)	(X)	(X)	(X)
C09	Remote management of device configurations	(X)		X	X	X	X	X
C10	Remote management of middleware			X	X	X	X	X
C11	Remote management of applications				X	X	X	X
C15	Disconnectable application with WEB UI					X	X	
C16	Disconnectable application using relational sync				X	X	X	X
C17	Disconnectable application using transactional messaging				X	X	X	X
C18	Disconnectable application using Web services				X	X	X	X

The Application pattern that best fits the required capabilities for the ITSO Insurance scenario is the Distributed Rich Client application pattern. Figure 8-16 on page 226 depicts the Distributed Rich Client application pattern used for the ITSO Insurance scenario.


Distributed Rich Client (DRC)
Presentation/ Application
synch/asynch disconnectable
Application
Figure 8-16 Distributed Rich Client application pattern used for ITSO scenario

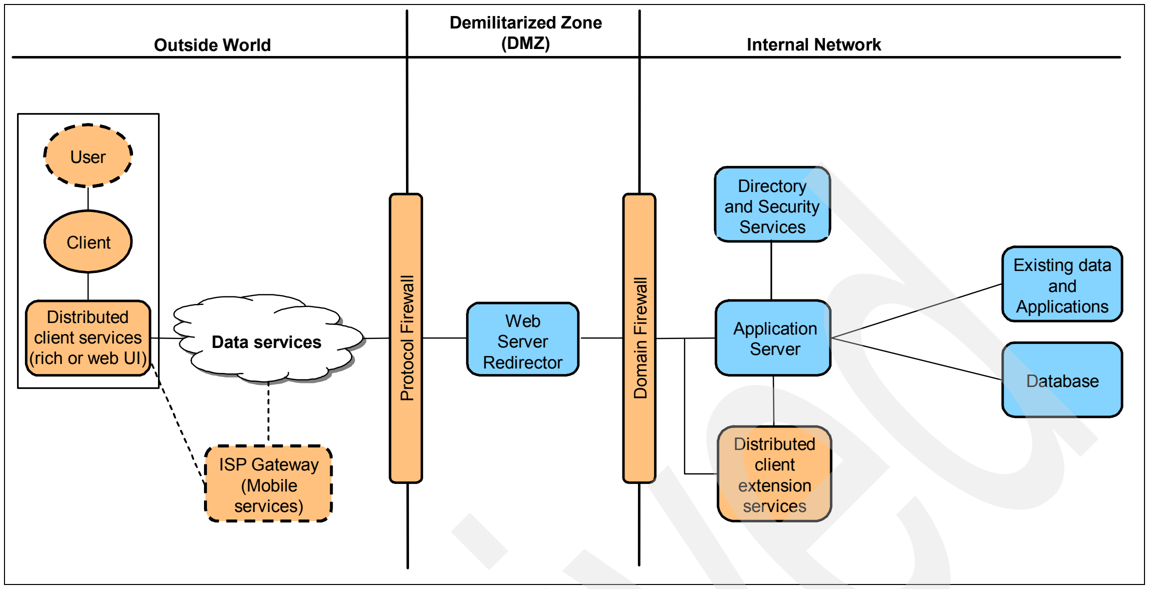
8.4.3 Select the Runtime pattern and Product mapping

Based on the Application pattern selected in 8.4.2, "Select the Application pattern" on page 224, we can now select the appropriate Runtime pattern and Product mapping.

Select the Runtime pattern

In our example, we selected the Distributed Rich Client application pattern based on the IT drivers and capabilities for the ITSO Insurance scenario. The Distributed Rich Client application pattern maps to the Distributed Rich Client::Runtime pattern.

The ITSO Insurance "Business requirements" on page 200 established that the Adjustor actor has a laptop client device, and as a requirement the solution needs to make use of the laptop. When referring to Table 6-2 on page 136, we selected the Distributed Rich Client runtime pattern since it fulfills the Application pattern capabilities and full function device (laptop) requirements. Figure 8-17 on page 227 depicts the Distributed Rich Client::Runtime pattern.



Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed client services (rich or web UI)
Data services
ISP Gateway (Mobile services)
Protocol Firewall
Web Server Redirector
Domain Firewall
Directory and Security Services
Application Server
Distributed client extension services
Existing data and Applications
Database

Figure 8-17 Distributed Rich Client::Runtime pattern

The Runtime pattern shows the logical components specific for running offline (disconnectable) clients with local application logic, data, and presentation layers. There are two major nodes to highlight in the Runtime pattern depicted in Figure 8-17:

- Distributed client extension services

  This node represents all services necessary for data synchronization, remote device and application management, transactional messaging, security, etc.

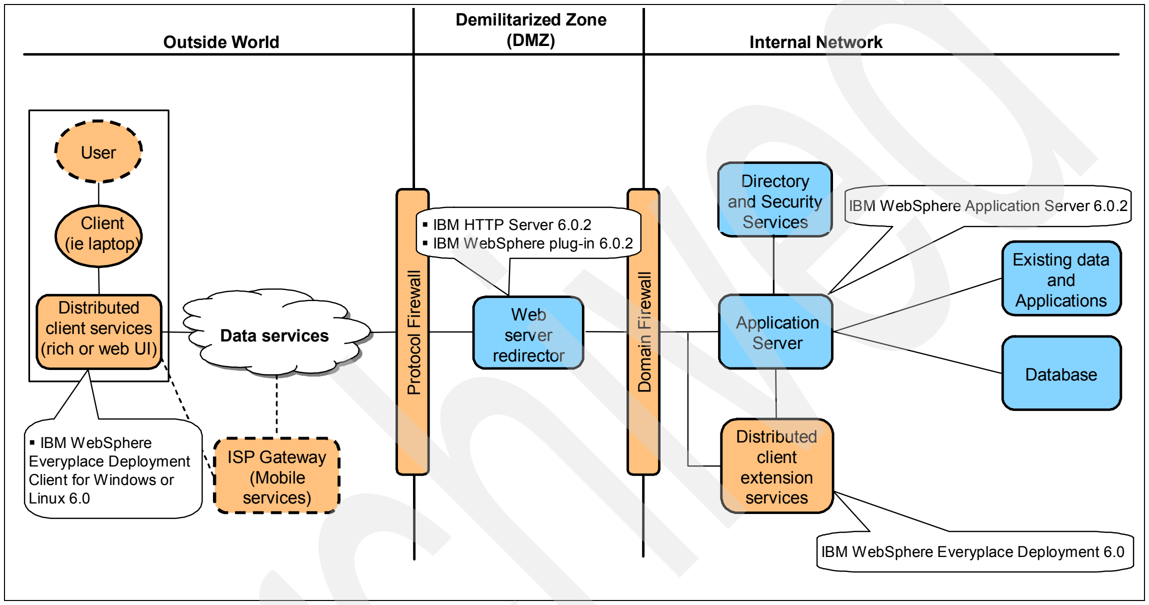
- Distributed client services

  This node contains those components that are designed to support device applications and interactions. The distributed client services node is linked very tightly with the distributed client extension services.

Distributed Rich Client::Runtime pattern::Product mapping

Once the Runtime pattern has been selected, there is a corresponding Product mapping that defines or instantiates the products used to fulfill the Runtime pattern. In our example, our Runtime pattern was fulfilled by the Distributed Rich Client::Runtime pattern::Product mapping, as seen in Figure 8-18 on page 228.

Note: In our example, the ITSO Insurance company required that we make use of the existing laptop systems used by adjusters. The laptop device criteria requires the use of WebSphere Everyplace Deployment Client for Windows and Linux to support the full function laptop device and required application capabilities. For this reason, we selected the Distributed Rich Client::Runtime pattern::Product mapping, as seen in Figure 8-18.



Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client (ie laptop)
Distributed client services (rich or web UI)
Data services
ISP Gateway (Mobile services)
▪ IBM WebSphere Everyplace Deployment Client for Windows or Linux 6.0
Protocol Firewall
▪ IBM HTTP Server 6.0.2
▪ IBM WebSphere plug-in 6.0.2
Web server redirector
Domain Firewall
Directory and Security Services
IBM WebSphere Application Server 6.0.2
Application Server
Existing data and Applications
Database
Distributed client extension services
IBM WebSphere Everyplace Deployment 6.0

Figure 8-18 Distributed Rich Client::Runtime pattern::Product mapping

Architecture components of WED 6.0 based solutions

A universal component model for solutions based on IBM WebSphere Everyplace Deployment V6.0 is described in detail in 7.5, “Distributed Rich Client::Runtime pattern” on page 169.

Decomposition of WED server products (services)

To provide a higher level of details for component mapping and application design, we will decompose the IBM WebSphere Everyplace Deployment V6.0 (server) products.

IBM WebSphere Everyplace Deployment V6.0 Server includes the following products (services):

- IBM DB2 Everyplace Sync Server
- IBM MQ Everyplace V 2.0.1

- IBM Device Manager V 1.8

The following software components are WebSphere Everyplace Deployment prerequisites:

- IBM WebSphere Application Server V6.0.2
- IBM DB2 Universal Database V8.2 + Fixpack 9a

WebSphere Everyplace Deployment is supported on Microsoft Windows 2003 Server, and optionally supports being configured with Microsoft Active Directory Server on a separate node.

Decomposition of WED client products (services)

To provide a higher level of details for component mapping and application design, we will decompose the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (client) products.

IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux client includes the following products (services):

- Eclipse V 3.0.2 with OSGi framework
- IBM MQ Everyplace V 2.0.1 client
- IBM DB2 Everyplace V 8.2.1

8.4.4 Component model

This section describes the component model for the ITSO working example.

Logical view of the application

The logical view (see Figure 8-19 on page 230) represents the functional view to all high-level logical components of the application design. The logical view has been developed to cover all functional requirements represented by use cases. The functional blocks in the logical view are defined in more detail in the component model found in the next.

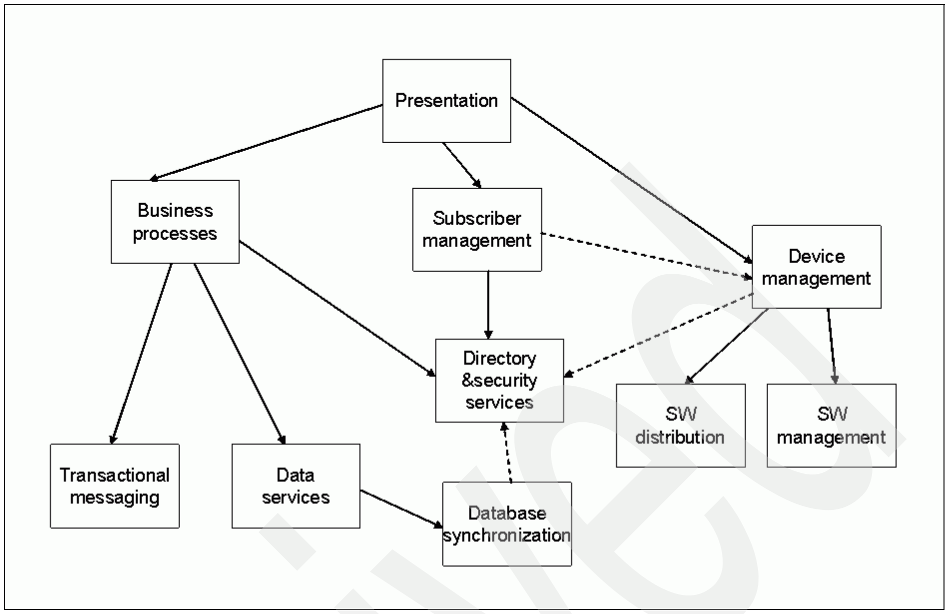


Figure 8-19 Logical view of application

Component model

This section describes the component model for the ITSO Insurance MobileAdjuster application. On this level we are focused only on components related directly to the service delivery. We have mapped them to conceptual nodes, which represent logical building blocks of the solution. We did not consider the operational aspects and operational related nodes and components (for example, firewalls, routers, http servers, proxies, etc.). We used the components described in “The Distributed client services and Distributed client extension services are instantiated by the following IBM products (see Figure 7-17 on page 173):” on page 172.

We considered the following three conceptual nodes relevant to our solution:

- The *MobileAdjuster server* represents the server part of the solution; in other words, it covers all services and capabilities required to connect the MobileAdjuster client to the existing back-end data and application as well as integrate the MobileAdjuster client to the existing management infrastructure.
- The *MobileAdjuster client* represents all components running on the user’s device necessary to enable the user to access the system remotely.
- *Existing back-end* in this model represents all existing systems, services, and data that need to be accessed or used by the remote user.

Figure 8-20 shows the model of the ITSO Insurance MobileAdjuster application, which is an example of the Distributed Rich Client application pattern.

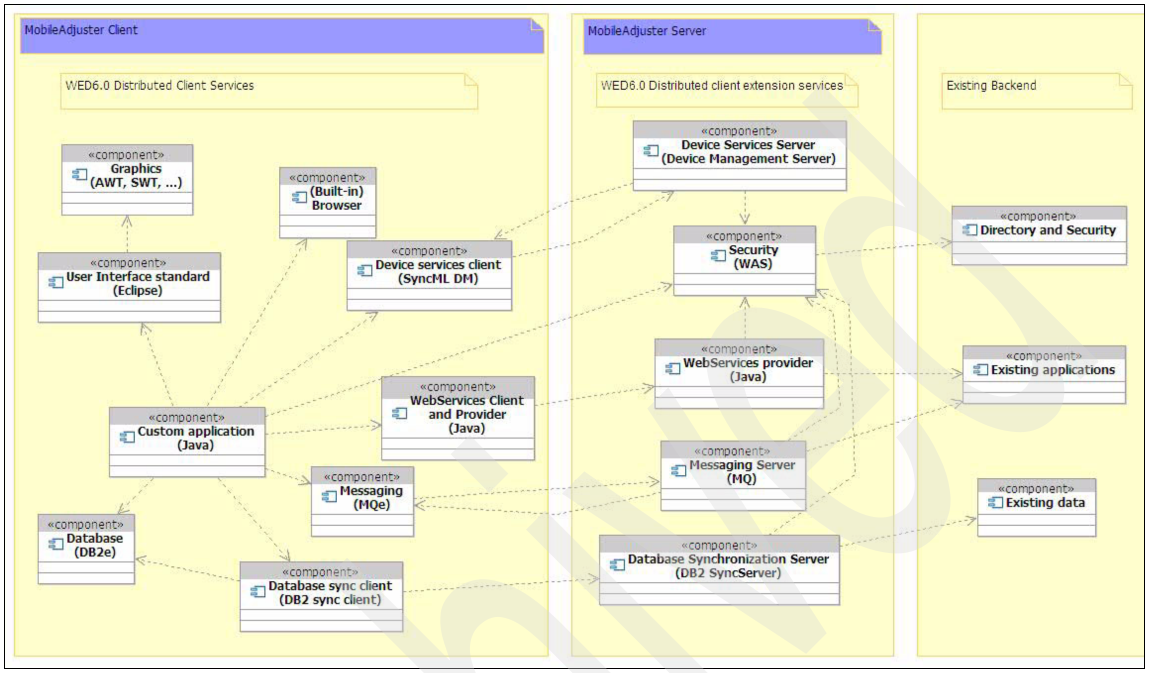

MobileAdjuster Client
WED6.0 Distributed Client Services
«component» Graphics (AWT, SWT, ...)
«component» (Built-in) Browser
«component» User Interface standard (Eclipse)
«component» Device services client (SyncML DM)
«component» WebServices Client and Provider (Java)
«component» Custom application (Java)
«component» Messaging (MQe)
«component» Database (DB2e)
«component» Database sync client (DB2 sync client)
MobileAdjuster Server
WED6.0 Distributed client extension services
«component» Device Services Server (Device Management Server)
«component» Security (WAS)
«component» WebServices provider (Java)
«component» Messaging Server (MQ)
«component» Database Synchronization Server (DB2 SyncServer)
Existing Backend
«component» Directory and Security
«component» Existing applications
«component» Existing data

Figure 8-20 MobileAdjuster component model

Figure 8-20 shows conceptual nodes of the solution design, including components and logical connections between these components. Figure 8-20 shows the logical connection between existing security system and mobile extension, and database connection, as well as integration with existing applications using transactional messaging and Web services.

Description of components

This section describes the server and client components depicted in Figure 8-20.

MobileAdjuster server components

The components are:

- *Security* provides a logical bridge to the existing common directory and security services and pervasive extension of security infrastructure. It manages the mobile subscriber database and provides binding for the device management server. On the other side, this component is responsible for managing the security device component running on the device and provides data for remote user authentication. The security component is responsible for holding pervasive subscriber-related data.

- The *Database Synchronization Server* is responsible for transferring data between the temporary database on the pervasive server and the existing back-end databases, and realizes physical synchronization of data between MobileAdjuster Server and MobileAdjuster Client devices. It processes data filtering, data transformation, and potential synchronization conflicts between the MobileAdjuster Client device and the MobileAdjuster Server. This component consolidates data inputs from all MobileAdjuster Client devices and provides a consolidated data set to the back-end data services connector for back-end synchronization.
- The *Messaging Server* component is responsible for realizing secure messaging services between MobileAdjuster Server and MobileAdjuster Client devices. In addition, this component is used to realize integration with back-end systems using transactional messaging.
- The *Web Services provider* component on the MobileAdjuster Server node realizes Web services based communication with the MobileAdjuster Client node and provides back-end integration with external systems using Web services.
- The *Device Services Server* component manages data relative to the user devices in terms of device inventory and SW management. To perform SW-related tasks it uses sub-component SW management and distribution. It holds all software-related data about managed devices. This component also stores a catalog of SW components and configurations available for supported devices. It performs software installation, update, and configuration remotely on devices.

MobileAdjuster client components

These components are:

- The *User Interface* component realizes a disconnectable local Web user interface running on the device. It provides user access to the system.
- The *Custom Application* component represents the business logic running on the user device and provides the user with the business logic necessary to work with data synchronized to the device. This component is also responsible for performing secure user authentication based on authentication data provided by the Security component of the MobileAdjuster Server. For this purpose this component holds data necessary to verify user authenticity and the access level defined in the back-end Directory and Security services. This component acts as the central security authority for local data access, data synchronization transactional messaging, and Web services based communication through the Custom Application component.

- The *Database* component provides the Device Custom Application component with access to data stored on the device. This component is responsible for encryption of locally stored data on the device.
- The *Database sync client* component realizes secure and reliable data synchronization between the MobileAdjuster Client and the MobileAdjuster Server.
- The *Messaging* component receives messages from the Messaging Server and makes them available for the device business logic component. This component is also responsible for sending messages to the server.
- The *Device Services client* component is the other part of device management and is responsible for performing tasks initiated by the Device Services Server including hardware and software related tasks (jobs). This component represents the *client platform*.

Existing data, *Directory and security services*, and *Existing applications* in this model represent only different entities for back-end integration. We do not consider cardinality or variations of these systems at this moment.

8.4.5 Where to go next

You have now completed the solution design for the ITSO Insurance scenario. The remaining chapters demonstrate how to implement the solution:

- Chapter 9, "Application design" on page 237
- Chapter 10, "Development environment installation" on page 255
- Chapter 11, "Application development" on page 279
- Chapter 12, "Runtime environment installation" on page 325
- Chapter 13, "Application deployment" on page 345
- Chapter 14, "Managing clients" on page 375
- Chapter 15, "Application walkthrough" on page 407

Part 3

ITSO MobileAdjuster patterns working example

This part demonstrates how to reuse the patterns for the fictitious ITSO Insurance scenario. The scenario highlights the Distributed Rich Client application pattern, which is used to extend the existing enterprise application with a SOA. The solution provides mobile adjusters with the capability to work on-site with the customer in a disconnectable mode, and sync data with the enterprise when connected to the network. The Runtime pattern is instantiated by IBM WebSphere Everyplace Deployment V6.0.

9

Application design

This chapter describes the application design for the ITSO working example defined in Chapter 8, “Requirements analysis and solution design” on page 197.

The chapter is organized into the following sections:

- Application design overview
- Import design artifacts into RSA
- Database design
- Use case realization and design components

9.1 Application design overview

The Rational Unified Process (RUP) employs a use-case driven approach, which means that use cases defined for a system are the basis for the entire development process. Use cases are realized in a design model as part of the RUP Analysis and Design discipline. Use-case realizations describe how the use case is supported by the design in terms of interacting objects in the design model.

In Chapter 8, “Requirements analysis and solution design” on page 197, we defined the requirements, use cases, and solution architecture. This chapter focuses on analyzing the behavior of the application and creating an initial set of elements that provide the appropriate behavior.

After the initial elements are identified, they are further refined. Design components produce a set of components that provide the appropriate behavior to satisfy the requirements on the system. If the system includes a database, then designing the database occurs in parallel. The result is an initial set of components, which are further refined in implementation.

This sample application is not meant to be a complete solution. It is meant to show the various technologies and demonstrate how to combine them into an application.

We focus on the mobile application running on a laptop used by the *Adjuster*. Therefore, we omit all the use cases that are associated with the actors *Administrator* and *Agent*.

We start with the analysis of the data residing in the back-end system and define the data to be synchronized to the mobile device. Next, we focus on the design of the client application, making use of following key mobile access services:

- Relational database synchronization
- Transactional messaging
- Web services

The client application is split up into the following functional areas:

- Mobile Adjuster

  This is the primary client-side application for capturing and synchronizing claims data and transmit messages.

- Image Manager

  This is used to capture and upload images via Web Services.

- Mobile Adjuster Configuration

This is used to configure the Mobile Adjuster with the proper DB2e, MQe, and Web services host.

9.2 Import design artifacts into RSA

We have provided the ITSO working example design artifacts in a Rational Software Architect project interchange file. To import the project interchange file, do the following:

1. Start Rational Software Architect.

   For information about installing Rational Software Architect, refer to Chapter 10, “Development environment installation” on page 255.
2. Open the Modeling perspective.
3. Select **File** → **Import**.
4. When the Import Wizard appears, select **Project Interchange** and then click **Next**.
5. In the Import projects dialog, browse to the C:\6775code\model folder and select **AccessIntegrationPatterns_ITSO_example.zip**. Click **Open**.
6. Click **Select All** and then click **Finish**.

9.3 Database design

As described in 8.1.1, “Initial context” on page 198, the ITSO Insurance company had an existing Web-based enterprise application with a back-end database. The Adjuster in the field only needs a subset of the information and the application, with the capability to synchronize the data to/from the client-server.

9.3.1 Analyze the host database

The ITSO Insurance company host database contains all the information about the clients, insurance policies, claims, and repair cost for parts.

In our scenario the host database (MOBADJ) has four tables (see Figure 9-1 on page 241):

- CLAIMS
- CUSTOMERS
- PARTS
- POLICIES

The CUSTOMERS table contains the following columns:

- customerid (primary key)
- name
- address
- phone
- policytype
- deductible
- expires

The table CLAIMS contains the following columns:

- claimid (primary key)
- adjusterid
- customerid
- claimdate
- claimitem
- damage
- repair
- replace
- status
- loanerneeded
- notes
- hasimage
- imagename

The table PARTS contains the columns:

- type (primary key)
- category
- description
- repaircost
- replacecost

The table POLICIES contains only two columns, policyitem and customerid. Together they are the compound key for this table. Customerid is also a foreign key from table CLAIMS.

The physical database design includes model elements (such as tables, views, and stored procedures) that represent the detailed physical structure of the database and model elements (such as schemas and tablespaces) that represent the underlying data storage design of the database.

Schema DB2ADMIN is the container for elements of the data model that represents the overall structure of the database. Used for managing security and ownership of tables.

Figure 9-1 displays the ITSO example MOBADJ database table definitions. The database table definitions were created within Rational Software Architect by creating a connection to the MOBADJ.

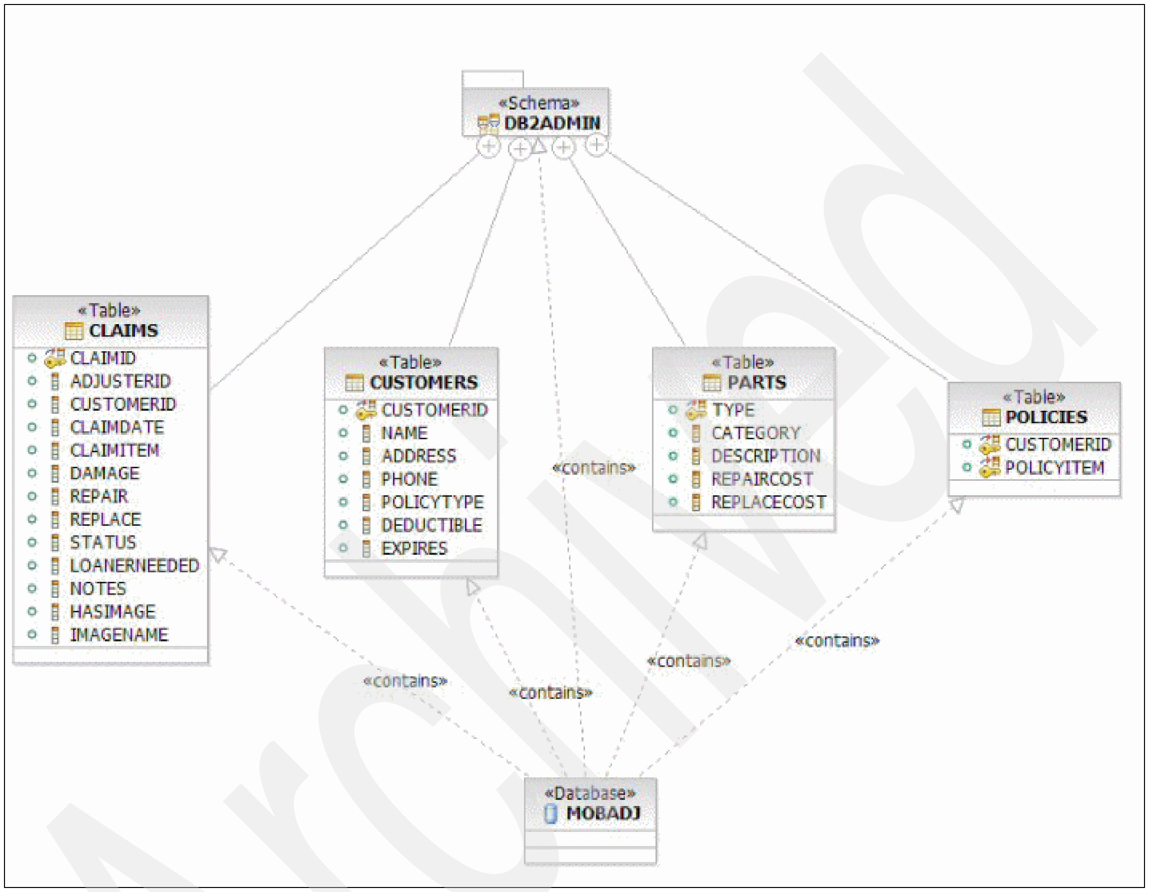


Figure 9-1 Mobile Adjuster Database table definition

9.3.2 Design mobile database

A major step for designing the mobile database is to decide which data from the back-end database should be available on the mobile device. Especially when a PDA is used, it should be reduced to the absolute minimum. Reducing the amount of data to a minimum is desired, especially when using devices such as PDAs that are more limited when compared to a laptop system.

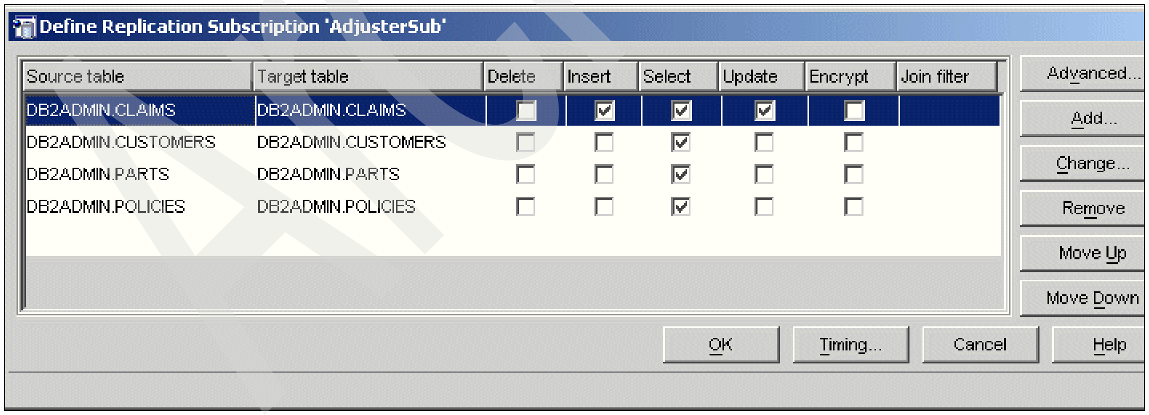
The design has direct impact on performance and security. Less tables and less rows result in faster synchronization times and less airtime cost (for example, GPRS or UMTS). Fewer rows in the mobile database also result in faster table scans. And partial data security can already be ensured by providing only the data that is necessary for the user to perform tasks.

In our example scenario each adjuster has one mobile device dedicated to them. To reduce the amount of data stored in the mobile database, we synchronize only the claims that have been assigned to the adjuster. As the adjuster should also be able to create new claims without prior assignment, we need all customer, policy, and part data on the device and cannot reduce it to the amount that is related to the claims.

Note: In a real scenario it would be necessary to define further filter criteria for customer, policy, and part data, as it would represent thousands of entries.

Especially when using a PDA, all the data would not fit on the device's storage and have high impact on table scan performance, and have security exposure.

Given the requirements, we need one subscription set *AdjusterSubSet* with one subscription *AdjusterSub*. The subscription set will be assigned to the user group DB2eAdjusters, which represents all adjusters, as all of them have the same role. The subscription contains the tables CLAIMS, CUSTOMERS, PARTS, and POLICIES. It will allow *insert*, *select*, and *update* operations for the table CLAIMS. For all other tables only *select* should be allowed, as the adjuster will not modify customer information or the policies and parts. Figure 9-2 shows how the subscription *AdjusterSub* is defined in the IBM DB2e Mobile Device Administration Center (MDAC).



Source table	Target table	Delete	Insert	Select	Update	Encrypt	Join filter
DB2ADMIN.CLAIMS	DB2ADMIN.CLAIMS	☐	✓	✓	✓	☐	
DB2ADMIN.CUSTOMERS	DB2ADMIN.CUSTOMERS	☐	☐	✓	☐	☐	
DB2ADMIN.PARTS	DB2ADMIN.PARTS	☐	☐	✓	☐	☐	
DB2ADMIN.POLICIES	DB2ADMIN.POLICIES	☐	☐	✓	☐	☐	

Figure 9-2 MDAC subscription definition

Since we only synchronize claims based on the adjuster's assignment, we define a horizontal filter[1] for the CLAIMS table.

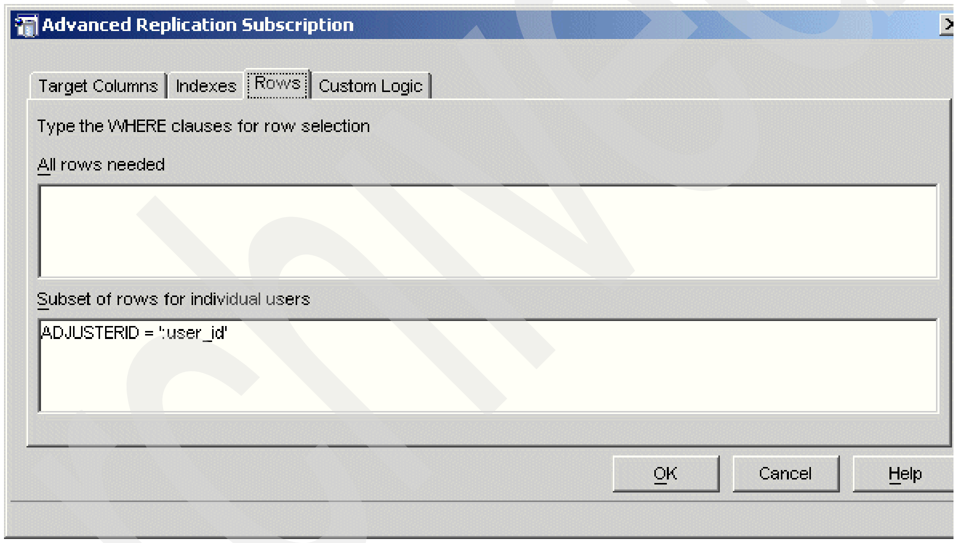
The ADJUSTER_ID column of the CLAIMS table correlates to the user names that are defined as users in the MDAC. The following SQL statement identifies the appropriate rows to filter using a filter parameter ':user_id'.

```
SELECT * FROM CLAIMS WHERE ADJUSTERID = ':user_id'
```

The parameter ':user_id' will be mapped on the group filter level to the sync server variable $USERNAME. When the user “adjuster1”, for example, synchronizes the data, the SyncServer will determine the user name and perform the customized SQL statement:

```
SELECT * FROM CLAIMS WHERE ADJUSTERID = 'adjuster1'
```

Figure 9-3 shows how this filter is defined in the MDAC and Figure 9-4 on page 244 shows the according group filter definition.



Type the WHERE clauses for row selection

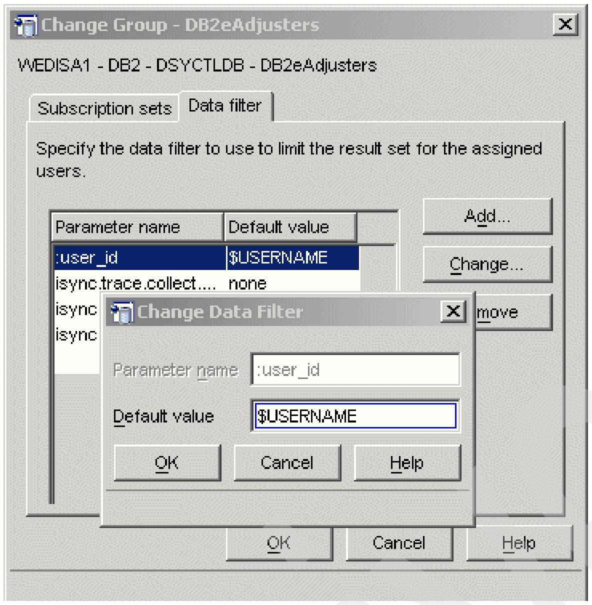
All rows needed

Subset of rows for individual users

ADJUSTERID = ':user_id'

Figure 9-3 MDAC horizontal filter definition

[1] The function of identifying and synchronizing a subset of rows of a source table to the corresponding table on a mobile device is called *horizontal filtering*.


Change Group - DB2eAdjusters

WEDISA1 - DB2 - DSYCTLDB - DB2eAdjusters

Specify the data filter to use to limit the result set for the assigned users.

Parameter name	Default value
:user_id	$USERNAME
isync.trace.collect....	none
isync	
isync	

Change Data Filter

Parameter name: :user_id

Default value: $USERNAME

Figure 9-4 MDAC group filter definition

Note: In the sample application you will find another table, *MESSAGES*, in the mobile database. This table is not synchronized and is only used to save the messages transferred via MQe locally.

9.4 Use case realization and design components

The purpose of a use-case realization is to define how the system implements a given use case. A use-case realization contains a domain diagram that describes the participants of the use case and their relationships, and a sequence diagram that describes the workflow of the use cases. Within Rational Software Architect this is known as a Process Helper.

We begin the use-case realization by analyzing the use cases defined in Chapter 8, “Requirements analysis and solution design” on page 197. We identify which access services and custom application components participate in each use case and to which artifact type they belong (for example, entity, boundary, or controller).

Note:

- *Entity* classes capture the data in the system.
- *Boundary* classes describe the interfaces between the system and the actors. If the actors are human, the boundary classes often describe the software interfaces that access the system. If the actors are systems, such as computers or other software applications, the boundary classes often represent software interfaces, such as APIs.
- *Controller* classes contain the implementation of the business rules and logic. They are purposefully abstracted from the user and are accessed only by the boundary classes or superior controller classes. Controller classes usually interact with one or more entity classes.

The second step is to create a sequence diagram that describes how these components interact.

The final result is a first draft of class diagrams with attributes and methods defined for each class. These classes will usually be further refined during the design and implementation iterations, but will not be described in this book.

Note: The design artifact details can be found in the Access Solution Example project after importing the AccessIntegrationPatterns_ITSO_example.zip project interchange file, as described in 9.2, “Import design artifacts into RSA” on page 239.

9.4.1 MobileAdjusterClient main entities

The domain model describes only the domain of the system, and does not describe any details of the application itself. It accomplishes this by modeling only the entity classes of the system and their relationships. The domain model diagram is called static, because it models only the structure of the system.

The entities represent data beans of the data that is persistent in the mobile database. The operations of each of the following entities should be modeled as a use-case and sequence diagram.

- The CustomerInfo Entity describes a customer’s insurance contract information. It has the main attributes customerid, name, address, phone, policytype, and expires.
- The ClaimInfo Entity describes a claim record. It has the main attributes claimID, adjusterID, customerID, claimDate, claimItem, damage, repair, status, loanerneeded, notes, image, and imagename.

- The PartInfo Entity describes the parts of the insured object. It has the main attributes type, category, description, repaircost, and replacecost.
- The Policy Entity describes the object insured. It has the main attributes customerid and policyItem.
- The MessageInfo Entity will be introduced later; it is a *helper* entity.

9.4.2 UCR_3-9 RetrieveSendClaimData

Use-case “UC_3 Retrieve new claims” and “UC_9 Sync claim data” are very similar and differ only in the direction of the data flow. Therefore both use cases are represented by one use-case realization <<UCR_3_9 RetrieveSendClaimData>>.

Participants

By analyzing the use-case activities, we identify the following participants and stereotype membership for this use case; see Table 9-1.

Table 9-1 UCR_3-9 RetrieveSendClaimData

Use-case activities	Use-case realization participants
1. Click Refresh data button.	► SyncForm (Boundary)
2. Data refresh process.	► AdjusterService (Controller) ► AdjusterDatabase (Controller) ► DatabaseSyncManager (Controller)
3. Synchronization status message.	► SyncForm (Boundary)

Description of the participants:

- SyncForm is a boundary and offers the sync button to the user, and presents the sync success message and the data record summary.
- AdjusterService is the controller and represents the application business logic.
- AdjusterDatabase is the controller for the DB2 access service component. It performs tasks like select, update, and modify on the mobile database, which contains claim, customer, policy, and part related information.
- DatabaseSyncmanager is the controller that triggers the synchronization of data with the back-end by using the DB2 database sync client component.

Figure 9-5 on page 247 shows the participants and their dependencies. Following the MVC concept, we add an additional controller AdjusterServlet to the ones identified above. This servlet is the central component to handle the requests and dispatch the actions to the model. The model is represented by the AdjusterService and the entities. The AdjusterService fetches data from lower levels or components, handles, the data and makes any necessary calculations. For example, it triggers the controllers of the access service components like AdjusterDatabase or DatabaseSyncmanager.

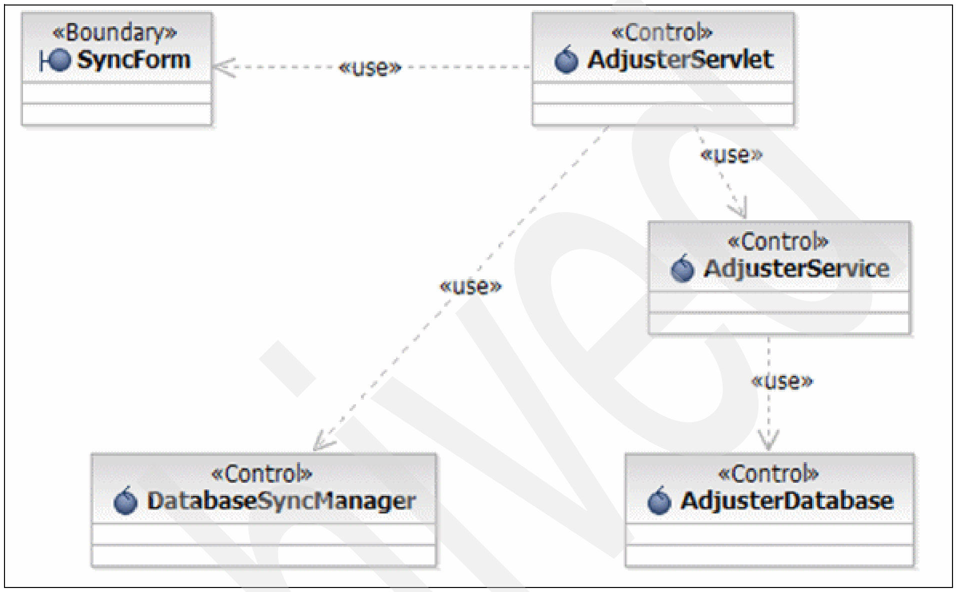


Figure 9-5 UCR_3-9RetrieveSendClaimData participants

Sequence diagram

When the user clicks the Refresh data button the SyncForm an event “sync” is sent to the AdjusterServlet. The AdjusterServlet calls the performSync operation in the AdjusterService that triggers the DatabaseSyncManager. The DatabaseSyncManager connects to the SyncServer and initiates the synchronization of the mobile database with the server database. When the synchronization is finished the sync status is sent back to the AdjusterServlet, and it forwards it for display to the SyncForm. The AdjusterServlet calls in addition the AdjusterDatabase to receive the updated data record summary. It contains the number of claims, customer, and part records in the database and is forwarded by the AdjusterServlet to the SyncForm for display.

Figure 9-6 on page 248 displays the UCR_3-9RetrieveSendClaimData sequence diagram.

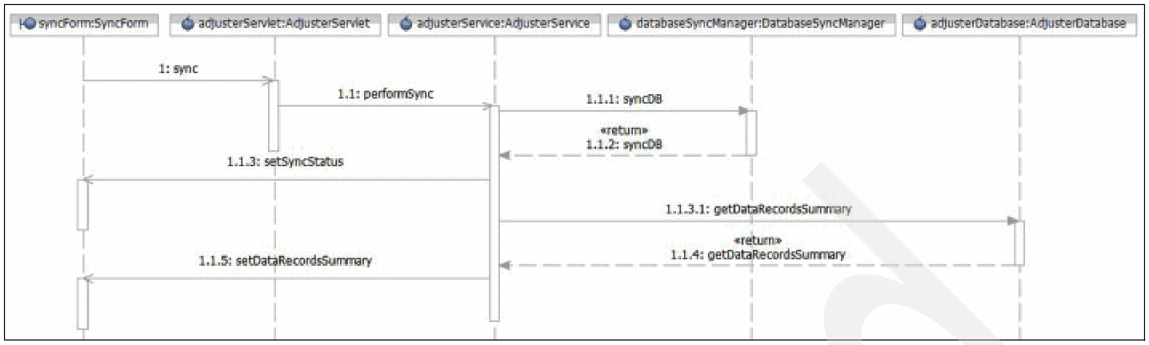

syncForm:SyncForm
adjusterServlet:AdjusterServlet
adjusterService:AdjusterService
databaseSyncManager:DatabaseSyncManager
adjusterDatabase:AdjusterDatabase
1: sync
1.1: performSync
1.1.1: syncDB
«return»
1.1.2: syncDB
1.1.3: setSyncStatus
1.1.3.1: getDataRecordsSummary
«return»
1.1.4: getDataRecordsSummary
1.1.5: setDataRecordsSummary

Figure 9-6 UCR_3-9RetrieveSendClaimData sequence diagram

9.4.3 UCR_4 Enter Claim Data

The use-case UC_4 Enter Claim Data is represented by one use-case realization UCR_4_EnterClaimData.

Participants

For this use case we identify the participants listed in Table 9-2. Some of them have already been identified for UCR_3-9 RetrieveSendClaimData. If they take on additional tasks, we will extend the description.

Table 9-2 UCR_4 EnterClaimData

Use-case activities	Use-case realization participants
1. Select claim from list.	► UrgentClaimList (Boundary) ► ClaimInfo (Entity) ► AdjusterDatabase (Controller)
2. Enter accident data.	► EditClaimForm(Boundary) ► ClaimInfo (Entity) ► CustomerInfo (Entity) ► PartInfo (Entity)
3. Attach image.	► EditClaimForm (Boundary) ► ImageRepository (Entity)
4. System calculates damage.	► AdjusterService (Controller)

Use-case activities	Use-case realization participants
5. Submit the claim for approval.	► ConfirmClaimForm (Boundary) ► AdjusterDatabase (Controller) ► BaseMQeTransport (Controller) ► MessageInfo (Entity)

The use-case realization participants are:

- *UrgentClaimList* is a boundary and shows the list of all urgent claims assigned for the adjuster.
- *EditClaimForm* is a boundary and presents the form to gather the claim information and a continue button.
- *ConfirmClaimForm* is a boundary that is used to display the claim data summary and to offer a submit button for the user.
- *ImageRepository* is an entity and issued to save the images of the claims.
- *ClaimInfo* is an entity that describes a claim record with the attributes claimID, adjusterID, customerID, claimDate, claimItem, damage, repair, status, loanerneeded, notes, image, and imagename. This data bean is made persistent in the mobile database table CLAIMS and temporarily used to gather the claim-related information and retrieve it.
- *CustomerInfo* is an entity that describes a customer's insurance contract information. It has the main attributes customerid, name, address, phone, policytype, and expires. It is used to temporarily store the client-related information after retrieving it from the mobile database table CUSTOMERS.
- *PartInfo* is an entity that describes the parts of an object insured. It has the main attributes type, category, description, repaircost, and replacecost. This data bean is used to temporarily store parts information after retrieving it from the mobile database table PARTS.
- *AdjusterService* is the controller and represents the application business logic. For this use case it handles the claim data and calculates the damage.
- *BaseMQeTransport* is the controller for the MQ messaging component. It sends the claim for approval to the back-end system.
- *MessageInfo* is an entity that describes the message. This message is the container for a claim when it is sent to the back-end for approval. It has the main attributes claimID and message and is made persistent in the mobile database in the table MESSAGES. The table is not synchronized via the DatabaseSyncManager and is only used as a local data store.

Figure 9-7 on page 250 displays the UCR_4EnterClaimData participants.

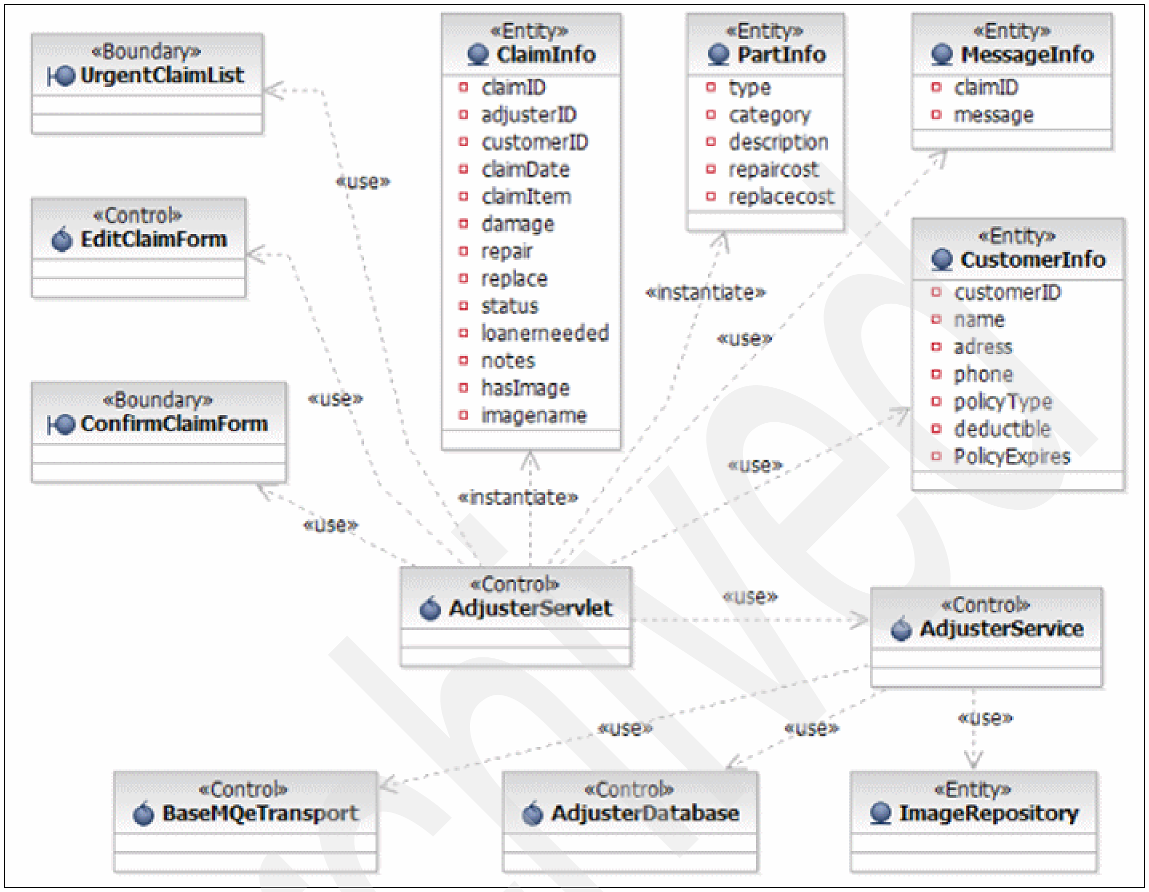


Figure 9-7 UCR_4EnterClaimData participants

Sequence diagram

Note: The sequence diagram was too large to fit within this book and still be readable. Please refer to the actual sequence diagram found within the imported sample code. Refer to “Import design artifacts into RSA” on page 239 for details.

When the user selects a claim from the UrgentClaimList for editing, it sends the event selEditClaim to the AdjusterServlet. The AdjusterServlet calls the getClaim method of the AdjusterService. The AdjusterService identifies the claim and calls the AdjusterDatabase to receive the complete claim record from the database. It determines, based on the claim record details, whether the claim contains an image. If it contains an image, the AdjusterService retrieves the image from the

ImageRepository. The combined result is passed back to the AdjusterServlet. The Adjuster Servlet calls in the next steps the AdjusterService to receive the customer details, claim status, and partinfos. All information is afterwards sent to the EditClaimForm to be displayed. The EditClaimForm sends the event “continue” to the AdjusterServlet when the user has entered the data and pressed the continue button. The AdjusterServlet collects the data input and forwards it to the AdjusterService for claiminfo validation. The AdjusterService computes the claim value by subtracting the customer’s individual deductible from the damagevalue entered in the EditClaimForm and sends the result back to the AdjusterServlet. The AdjusterServlet sends the validated claim information to the ConfirmClaimForm for display. The ConfirmClaimForm sends the event “submitClaim” to the AdjusterServlet when the user has pressed the submit button. The AdjusterServlet calls again the AdjusterService for further processing. If the claim contains an image, it will be saved in the ImageRepository. Finally, the AdjusterService calls the AdjusterDatabase to save the claim record in the mobile database and calls the BaseMQeTransport to send the claim for approval to the back-end system.

9.4.4 UCR_6 Upload image

The use case UC_6 Enter Claim Data is represented by one use-case realization UCR_6_EnterClaimData.

The images do not belong to the core data of a claim record. A claim can be processed without having an image. Therefore the Web service is used to provide an independent way for sending the image data to the server.

The claim record data does not create much traffic. Images would increase the traffic dramatically, extending synchronization times. Images can be sent while having a fast connection, independent from the claim records.

The location of images is synchronized via the claim record. The client-side application uses the ImageServlet to upload the image to the server-side application, which stores the image on the server file system. The uploadImage calls to the Web service to upload the image to the server.

Participants

Based on the use-case activities, we identified the following participants; see Table 9-3.

Table 9-3 UCR_6 Upload image

Use-case activities	Use-case realization participants
1. Select image to be uploaded.	► ImageManagerForm (Boundary)

Use-case activities	Use-case realization participants
2. Application uploads image to the server.	► AdjusterService (Controller) ► WebService (Boundary)
3. Application sets image upload status.	► ImageManagerForm (Boundary) ► AdjusterService (Controller) ► ClaimInfo (Entity) ► AdjusterDatabase (Controller)

The use-case realization participants are:

- *ImageManagerForm* is a boundary and the user interface that presents all images of claim records.
- *WebService* is a boundary.
- *AdjusterService* is the controller.

Figure 9-8 displays the UCR_6UploadImage participants.

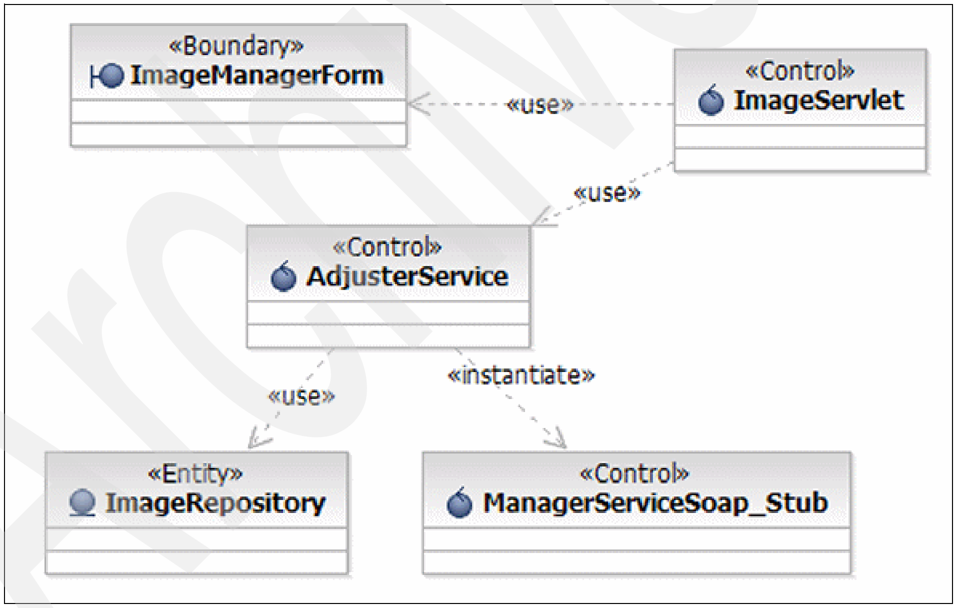


Figure 9-8 UCR_6UploadImage participants

Sequence diagram

The UCR_6UploadImage sequence diagram is displayed in Figure 9-9 on page 253.

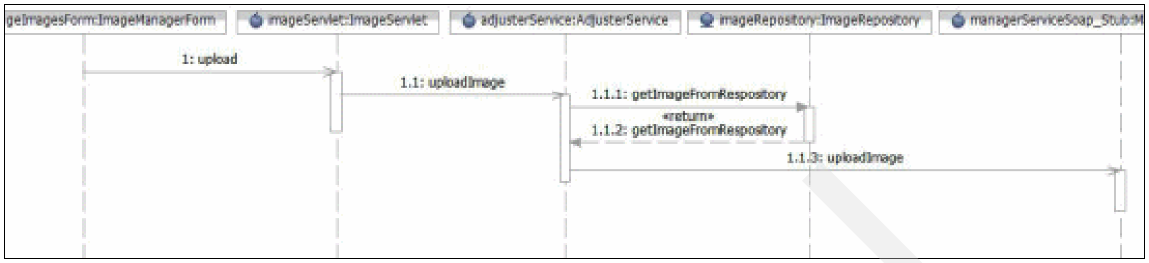

geImagesForm:ImageManagerForm
imageServlet:ImageServlet
adjusterService:AdjusterService
imageRepository:ImageRepository
managerServiceSoap_Stub:M
1: upload
1.1: uploadImage
1.1.1: getImageFromRespository
«return»
1.1.2: getImageFromRespository
1.1.3: uploadImage

Figure 9-9 UCR_6UploadImage sequence diagram

9.4.5 UCR_5 Process claim

The use case UC_5 Process Claim is represented by one use-case realization UCR_5_ProcessClaim. This use case belongs to the functional area Server and interacts with UC_4 Enter Claim Data.

Table 9-4 UCR_5 Process claim

Use-case activities	Use-case realization participants
1. Receive new claim notification.	► MQeListener(Boundary) ► MQeTransport (Controller)
2. Process the claim according to claim value.	► BusinessRuleProcessor (Controller)
3. System approves or rejects claim.	► BusinessRuleProcessor(Controller) ► AdjusterService (Controller) ► ClaimInfo (Entity) ► AdjusterDatabase (Controller)
4. System sends claim approval or rejection message.	► MQeTransport(Controller)

This component does not have a user interface; therefore we do not follow the MVC concept. The main controller triggers the MQeTransport/MQeListener controller.

In our example we have only one business rule. All claims with a value above 500 dollars will be rejected. All others will be automatically approved.

Claims are submitted to a queue. When a connection is available the claims are processed and the server receives the messages. If a device is disconnected, data stays in the queue but is not triggered automatically again after the first try. It is triggered by clicking “refresh messages” or by creating a new claim.

10

Development environment installation

The objective of this chapter is to describe how to implement the development environment used to develop and test the ITSO MobileAdjuster working example application. The procedures in this chapter are high level in some cases and intended to be used in conjunction with the product installation guides.

The value add of the procedures found in this chapter are three fold. First, we provide high-level installation and configuration procedures of the software components by node for our scenario. Second, the procedure includes sample values that put in context the information found in the product installation guides. Third, the procedures include best practices, tips, and workarounds based on our first-hand experiences.

The chapter is organized into the following sections:

- Introduction to the development tooling
- Scenario and planning overview
- Access Server node
- Developer node
- Import the completed sample code
- Create client configuration and run application

10.1 Introduction to the development tooling

This section is intended to highlight the development tooling used to develop client-side applications supported by WebSphere Everyplace Deployment.

IBM WebSphere Everyplace Deployment V6.0 supports the following client platforms:

- IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux

  This client platform is targeted at laptop and desktop clients.
- IBM Workplace Client Technology, Micro Edition V5.7.1, 5.7.2 (WCTME)

  This client platform is targeted at PDA type devices such as the PocketPC and Palm.
- IBM Workplace Client Technology, Micro Edition Enterprise Offering V5.8.1 (WCTME-EO)

  This client platform is targeted at laptop and desktop clients, and is essentially replaced by IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux.

10.1.1 Development tooling by client platform

Although there are three possible client platforms (WED, WCTME, WCTME-EO), we limit our discussion on the tooling to WED and WCTME.

Tooling for WED client-side applications

The IBM WebSphere Everyplace Deployment V6.0 product includes the WebSphere Everyplace Client Toolkit V6.0 used to develop client-side applications to be deployed to IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (clients). The WebSphere Everyplace Client Toolkit can be installed on one of the following IBM Rational Software Developer V6 products:

- IBM Rational Web Developer V6.0.0.1 (RWD) or later
- IBM Rational Application Developer V6.0.0.1 (RAD) or later
- IBM Rational Software Architect V6.0.0.1 (RSA) or later

The IBM Rational Software Developer products listed are in order of least to most capable. For example, RSA includes all enterprise development capabilities of RAD, but also includes modeling and design tooling used by architects.

Note: For more information about using WebSphere Everyplace Client Toolkit refer to *WebSphere Everyplace Deployment V6.0 - Client Application Development*, SG24-7183.

Tooling for WCTME client-side applications

IBM Workplace Client Technology, Micro Edition V5.7.x (5.7.1, 5.7.2) is supported by WebSphere Everyplace Deployment, but not included in the product packaging. To develop applications to be deployed on Workplace Client Technology, Micro Edition you will need IBM WebSphere Studio Application Developer V5.1 and the IBM Everyplace Device Developer V5.1 Toolkit.

Note: For more information about development for WCTME refer to *IBM Workplace Client Technology Micro Edition Version 5.7.1: Application Development and Case Study*, SG24-6496.

10.1.2 WebSphere Everyplace Client Toolkit overview

The objective of this section is to highlight the tooling (wizards, APIs, client test environment, samples, etc.) provided by the WebSphere Everyplace Client Toolkit V6.0 for developing client-side applications to be deployed to IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux.

The IBM WebSphere Everyplace Client Toolkit provides the tools necessary to create and test OSGi bundles, Web Applications, and Embedded Transaction Applications for use on theWebSphere Everyplace Deployment platform. The WebSphere Everyplace Client toolkit provides wizards that enable developers to create Client Services projects to develop client applications, without requiring them to become OSGi bundle internal experts.

- Client Services Embedded Transaction Project - Develop and deploy EJBs.
- Client Services Fragment Project - Develop fragments.
- Client Services Project - Develop rich client GUI applications messaging applications, database applications, and application components (plug-in/bundle).
- Client Services Web Project - Develop and deploy Web applications.
- Convert Project to Client Service Project - Convert a Java project to a Client Services Project.
- Migrate Extension Services Project - Migrate a WSDD 5.7 Extension Services project to a Client Services project.
- Platform Builder Project - Build a custom client platform.

- Mobile Web Services - Develop a Web Services client and/or provider and support security.

Figure 10-1 displays some of the wizards provided by the WebSphere Everyplace Client Toolkit.

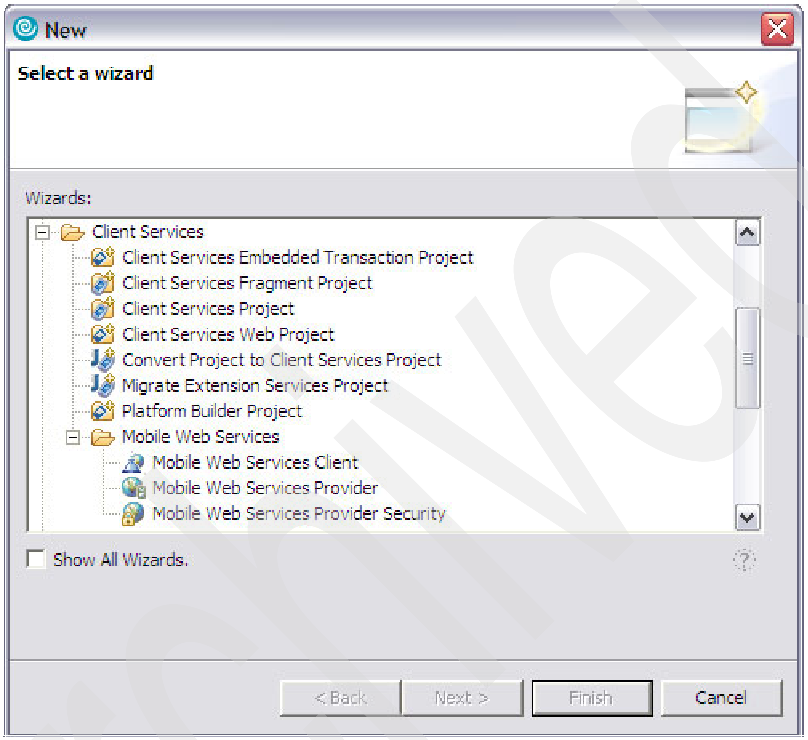

New
Select a wizard
Wizards:
Client Services
Client Services Embedded Transaction Project
Client Services Fragment Project
Client Services Project
Client Services Web Project
Convert Project to Client Services Project
Migrate Extension Services Project
Platform Builder Project
Mobile Web Services
Mobile Web Services Client
Mobile Web Services Provider
Mobile Web Services Provider Security
Show All Wizards.
< Back
Next >
Finish
Cancel

Figure 10-1 IBM WebSphere Everyplace Client Toolkit wizards

10.2 Scenario and planning overview

This section describes the development environment topology and solution components used to develop and test the ITSO MobileAdjuster application. In addition, we have included information about the hardware and software used to implement the development environment.

The development environment consists of the following two nodes depicted in Figure 10-2 on page 259:

- Developer node

This node includes the IBM Rational Software Architect development tooling and IBM WebSphere Everyplace Client Toolkit, which is used to develop and test the client-side applications.

- Access Server node

  This node is used to host the server-side application components and enable end-to-end testing within the development environment.

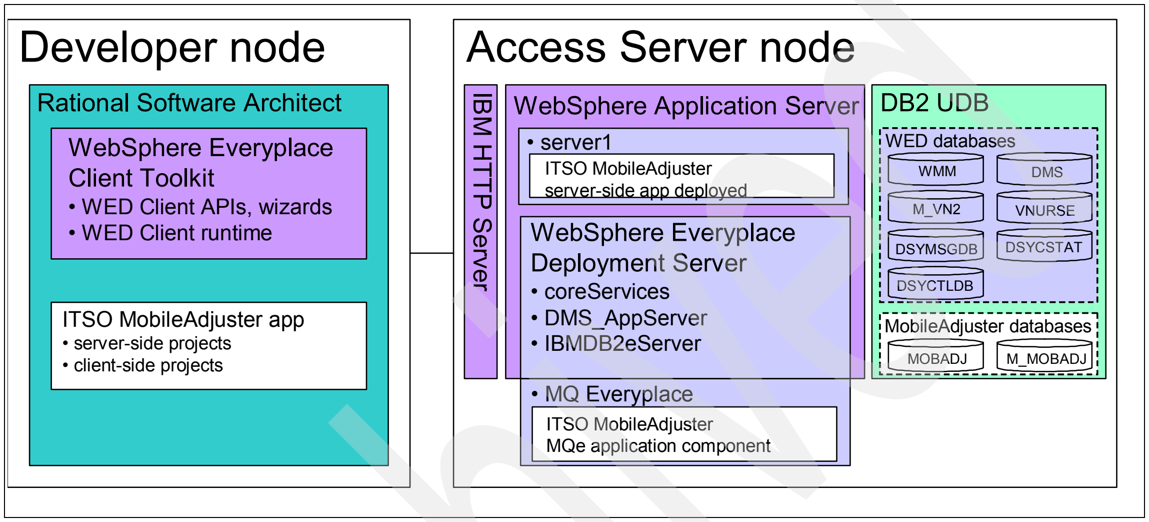


Figure 10-2 ITSO working example Development environment Product mapping

10.2.1 Hardware and software prerequisites

For detailed and official product information about the hardware and software requirements, we recommend that you refer to the IBM WebSphere Everyplace Client Toolkit V6.0 Release notes (<wect_cd_root>/docs/en/WCTReadme.html).

10.2.2 Hardware used within the ITSO runtime environment

We used the following hardware to implement the nodes of the ITSO development environment:

- Developer node
  - IBM ThinkCentre
    - 1 CPU, Intel® Pentium® 4 2.8 GHz
    - 4 GB RAM (2 GB used by host, and 2 GB used by VMWare)
    - 40 GB Hard Disk
    - 1 Ethernet Adapter
    - Hostname: radwin1.itso.ral.ibm.com

- Access Server node (VMWare image)
  - 1 CPU, Intel Pentium 4 2.8 GHz
  - 2 GB RAM
  - 10 GB Hard Disk
  - 1 Ethernet Adapter
  - Hostname: wedswin1.itso.ral.ibm.com

Tip: VMWare Workstation V5.0

In our example development environment, the Developer node software (Rational Software Architect and WebSphere Everyplace Client Toolkit) is installed on the host system, and the Access Server node is installed in a VMWare Workstation V5.0 - Windows 2003 Server image running on the same physical node.

10.2.3 Software used within the ITSO runtime environment

The ITSO working example was implemented using the software levels listed in each of the following tables by node.

Table 10-1 Developer node

Software	Version
Microsoft Windows XP Professional	XP + SP1 + critical fixes
IBM Rational Software Architect	6.0.1 **Note**: 6.0 + 6.0.1 Refresh
IBM DB2 UDB, Enterprise Server Edition	8.1.9.917 **Note**: 8.2 + Fixpack 9a
IBM Everyplace Client Toolkit **Note:** This includes the WebSphere Everyplace Deployment Client.	6.0

Table 10-2 Access Server node

Software	Version
Microsoft Windows 2003 Server	2003 + critical fixes
IBM WebSphere Application Server	6.0.2 **Note**: 6.0 + 6.0.2 Refresh
IBM DB2 UDB, Enterprise Server Edition	8.1.9.917 **Note**: 8.2 + Fixpack 9a
IBM WebSphere Everyplace Deployment	6.0

10.3 Access Server node

In order to do full testing for WebSphere Everyplace Deployment applications that include server and client application components, you will need a WebSphere Everyplace Deployment Server on a separate node from the Developer node.

For more detailed information about installing the WebSphere Everyplace Deployment Server, refer to 12.2, “Access Server node installation” on page 329, found in the Appendix C, “Rational Unified Process (RUP) overview” on page 483, which shares a common procedure.

10.4 Developer node

The Developer node contains the tooling to develop the application artifacts for the Distributed Rich Client application pattern. The primary tooling used for this purpose is IBM Rational Software Development Platform (Rational Software Architect, or Application Developer, Web Developer) and the WebSphere Everyplace Client Toolkit.

Complete the following tasks to install the software components of the Developer node:

- Windows XP installation
- Rational Software Architect installation
- WebSphere Everyplace Client Toolkit installation

10.4.1 Windows XP installation

For our example, we installed Microsoft Windows XP Professional, Service Pack 1 and critical fixes.

10.4.2 Rational Software Architect installation

The IBM WebSphere Everyplace Client Toolkit V6.0 requires the following prerequisite IBM Rational Developer products based on the IBM Rational Software Development Platform (SDP):

- IBM Rational Web Developer V6.0.0.1
- IBM Rational Application Developer V6.0.0.1
- IBM Rational Software Architect V6.0.0.1

At the time of writing this book, we tested with both IBM Rational Application Developer V6.0.1 (Rational Application Developer V6.0 with Refresh Pack

V6.0.1) and IBM Rational Software Architect V6.0.1 (Rational Software Architect V6.0 with Refresh Pack V6.0.1).

The following procedure outlines how to install Rational Software Architect, and can be used as a guide for Rational Application Developer as well.

Installation considerations

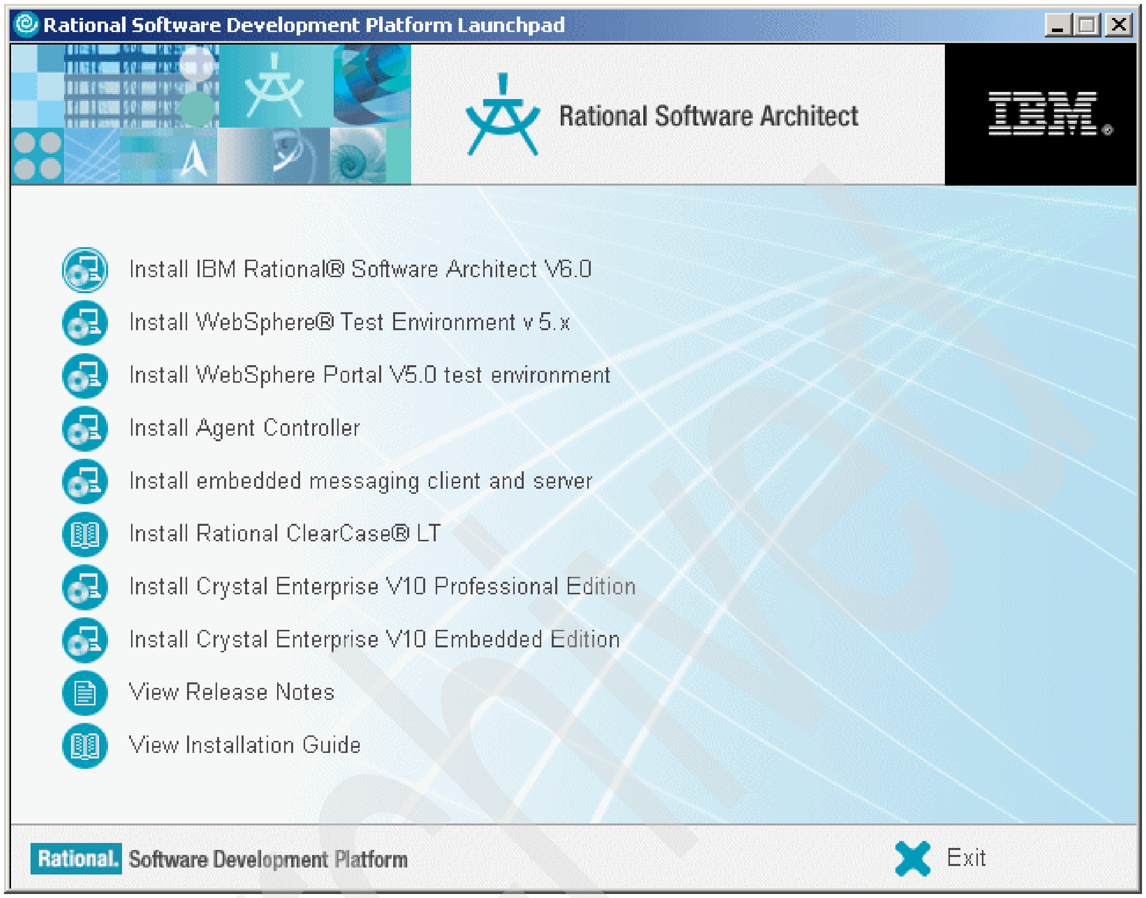
Prior to installing Rational Software Architect, beware of the following installation considerations:

- UNC network shares: Do not install Rational Software Architect from a UNC network share (for example, \\server\shareA). Instead, map the network drive to a drive letter (for example, net use x: \\server\shareA) so that the Rational Software Architect installer works properly.
- Standardize installation path for team development: Standardize the Rational Software Architect installation path for your development team. We found that many files within the projects have absolute paths based on the installation path; thus, when you import projects from a team repository such as CVS or ClearCase® you will get many errors.

Rational Software Architect V6.0 installation

To install IBM Rational Software Architect V6.0, do the following:

1. Start the Rational Software Architect Installer by running launchpad.exe from CD 1.
2. The IBM Rational Software Architect V6.0 components have separate installations from the main Launchpad Base page, as seen in Figure 10-3 on page 263.


Rational Software Development Platform Launchpad

Rational Software Architect

Install IBM Rational® Software Architect V6.0

Install WebSphere® Test Environment v 5.x

Install WebSphere Portal V5.0 test environment

Install Agent Controller

Install embedded messaging client and server

Install Rational ClearCase® LT

Install Crystal Enterprise V10 Professional Edition

Install Crystal Enterprise V10 Embedded Edition

View Release Notes

View Installation Guide

Rational. Software Development Platform

Exit

Figure 10-3 IBM Rational Software Architect V6.0 installation components

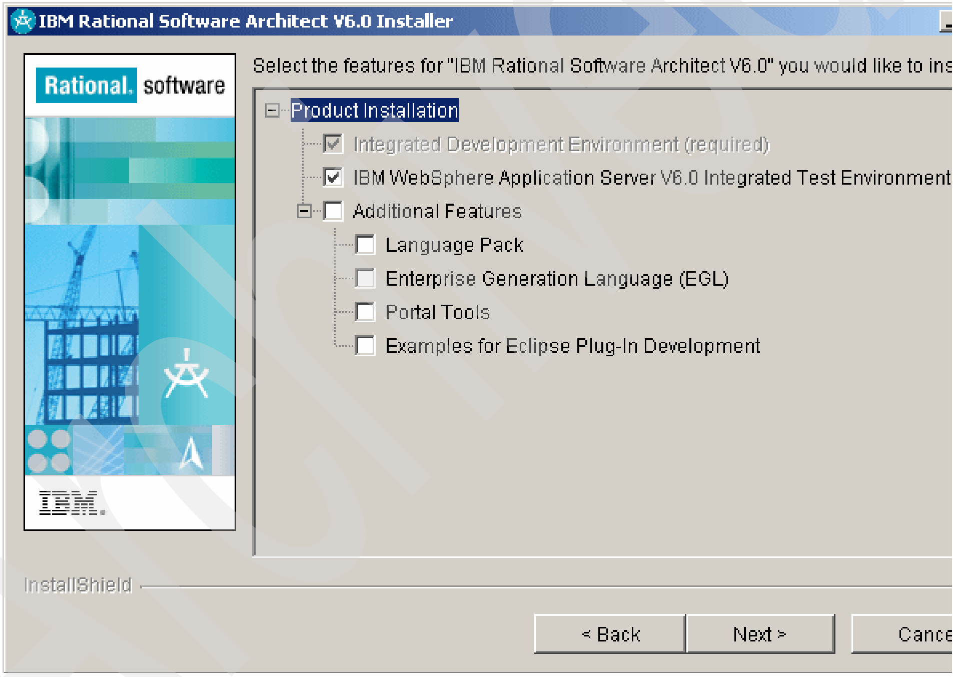
> **Note:** In our Development environment, we only need to select the Install IBM Rational Software Architect V6.0 component.

3. When the Welcome window appears, click **Next**.
4. When the License Agreement window appears, review the terms and if in agreement select **I accept the terms of the license agreement**. Click **Next**.
5. When the Install Directory window appears, we accepted the default C:\Program Files\IBM\Rational\SDP\6.0 and clicked **Next**.

Important: It is very important that you understand the implication of changing the default installation path. For example, if you plan on team development, we strongly recommend that all developers use the same installation path (such as default); otherwise you will run into problems.

We found that many files within the projects have absolute paths based on the installation path; thus, when you import projects from a team repository such as CVS or ClearCase you will get many errors.

6. When the Select Features window appears, select the appropriate features for your environment. For example, we selected the features displayed in Figure 10-4 for the book.

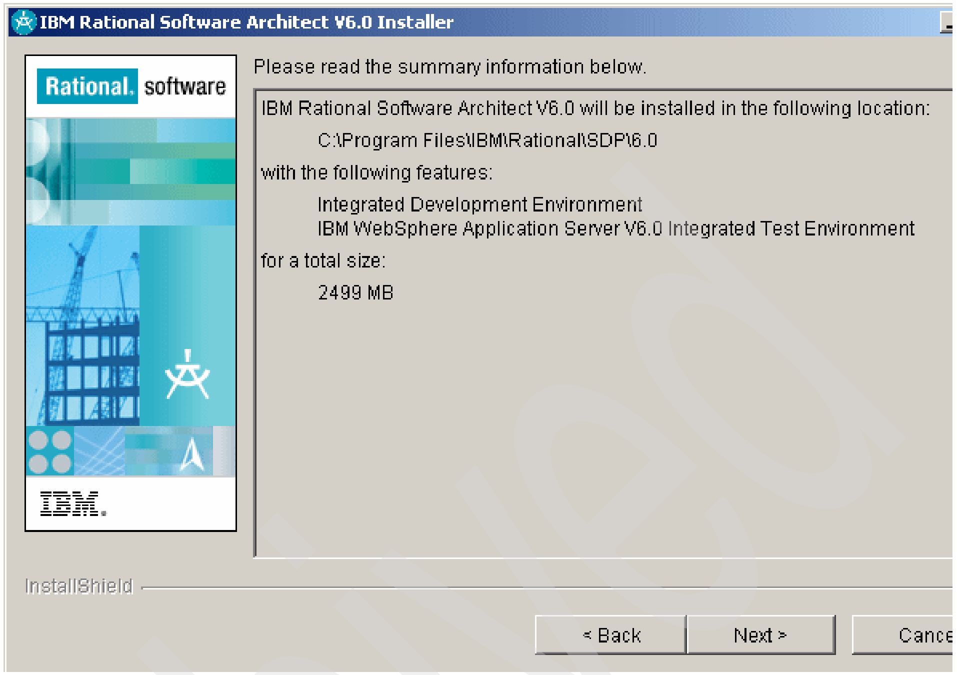


Select the features for "IBM Rational Software Architect V6.0" you would like to ins

- Product Installation
  - [✓] Integrated Development Environment (required)
  - [✓] IBM WebSphere Application Server V6.0 Integrated Test Environment
  - [ ] Additional Features
    - [ ] Language Pack
    - [ ] Enterprise Generation Language (EGL)
    - [ ] Portal Tools
    - [ ] Examples for Eclipse Plug-In Development

Figure 10-4 Install IBM Rational Software Architect V6.0 features

7. When the Installation Summary window appears, review your selections and click **Next** to begin copying files, as seen in Figure 10-5 on page 265.



Please read the summary information below.

IBM Rational Software Architect V6.0 will be installed in the following location:

C:\Program Files\IBM\Rational\SDP\6.0

with the following features:

Integrated Development Environment
IBM WebSphere Application Server V6.0 Integrated Test Environment

for a total size:

2499 MB

Figure 10-5 IBM Rational Software Architect V6.0 install summary

The installation can take 30 minutes to 2 hours depending on the processing speed of your system and the features selected.

8. When you see the message `The Installation Wizard has successfully installed IBM Rational Software Architect V6.0`, click **Next**.
9. The installation is now complete. We unchecked the “Launch Agent Controller install” and then clicked **Finish**. We will install the IBM Rational Agent Controller separately for profiling and testing purposes.

10.4.3 Rational Product Updater - Refresh Pack V6.0.1

The *Rational Product Updater* tool is used to apply fixes to IBM Software Development tooling. In this case, we install the Refresh Pack V6.0.1.

The following procedure describes how to install Refresh Pack V6.0.1 using the Rational Product Updater:

1. Ensure that Rational Software Architect is closed and test servers are stopped.
2. Start the Rational Product Updater by clicking **Start** → **Programs** → **IBM Rational** → **Rational Product Updater**.

3. Click **Find Updates**.
4. You may be prompted that there are required updates for the Product Updater. For example, in our case we have just installed IBM Rational Software Architect V6.0 and are prompted to install the IBM Rational Software Architect Refresh Pack V6.0.1. Click **Continue**.

   Tip: If the Rational Product Updater requires an update, you are prompted to install it before you can continue. The Rational Product Updater installs the update, restarts, and retrieves a list of available updates.

5. The IBM Rational Product Updater should be populated with updated fix information. Ensure that "Refresh Pack V6.0.1" and "Interim Fix 001 for Rational Software Architect V6.0.1" are checked. Click **Install Updates**.

   Tip: Detailed information about the fix is displayed in the right-hand window by selecting Refresh or Fix.

6. When prompted, click **Continue**.
7. When the License Agreement dialog appears, if in agreement select **I accept the terms in the license agreements**, and then click **OK** to begin the installation.

   Depending on the speed of your computer processor, the amount of RAM, and the speed of your Internet connection, the update might take an extended period of time to download and install.
8. After the installation is complete, click the **Installed Products** tab to verify that the Interim Fixes were installed successfully.
9. Close the Rational Product Updater.

Verify Rational Software Architect V6.0.1

After applying the Refresh Pack V6.0.1, we recommend that you take some steps to ensure that Rational Software Architect is working properly.

- Start Rational Software Architect.
- Ensure the WebSphere Application Server v6 test server starts properly.

10.4.4 WebSphere Everyplace Client Toolkit installation

This section describes how to install, configure, and verify the WebSphere Everyplace Client Toolkit.

Install WebSphere Everyplace Client Toolkit

To install the IBM WebSphere Everyplace Client Toolkit V6.0, do the following:

1. Review release notes for system requirements and product information, such as software limitations at the time of this release.
2. Log on to your system with an administrator ID that has write permission to the installation location and start the Rational Software Architect.
3. From the main menu, click **Help** → **Software Updates** → **Find and Install**. The Install/Update wizard is displayed.
4. Select **Search for new features to install**, and click **Next**.
5. When the Update Sites to visit window appears, click **New Local Site**.
6. Browse to select the drive and directory where the IBM WebSphere Everyplace Client Toolkit feature is located and then click **OK**.

   For example, if installing from the product CD, select the **WEC_Toolkit** directory and click **OK**.
7. Check the check box next to the site name (for example, D:\WEC_Toolkit), and then click **Next**.

   Note: The selection of the "Ignore features is not applicable to this environment" check box does not affect toolkit installation.
8. When the Search Results dialog appears, check **IBM WebSphere Everyplace Client Toolkit**, and then click **Next**.
9. When the Feature License window appears, review the terms of the license and, if in agreement, select **I accept the terms of the agreement**, and then click **Next**.
10. When the Install Location dialog appears, click **Finish** to begin the installation.

    The default installation directory is as follows:

    ```
    C:\Program Files\IBM\Rational\SDP\6.0\sdpisv\eclipse
    ```
11. When the Feature verification window appears, click **Install**.
12. When installation completes, you are prompted to restart the workbench for changes to take effect. Click **Yes** to continue.

    Note: Clicking **Apply Changes** does not properly configure the environment.

13. After the workbench restart, you will be prompted to set the preferences for WebSphere Everyplace Client Toolkit. Click **OK**.

    You will see a dialog that the bundles must be registered for WebSphere Everyplace Client Toolkit to function properly.

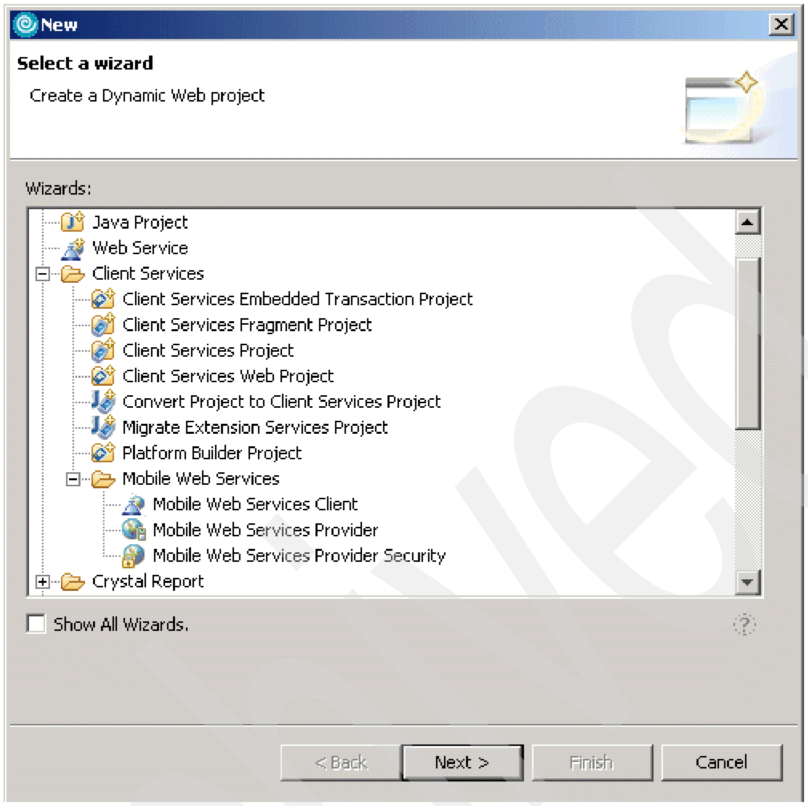
14. Restart the workbench.

> **Note:** When WebSphere Everyplace Client Toolkit registers the bundles it also inserts data into the Cloudscape database. The optimization function of Cloudscape does not take effect on the updated database tables until Rational Software Architect has been restarted (restarts Cloudscape).

Verify the WebSphere Everyplace Client Toolkit installation

To verify the WebSphere Everyplace Client Toolkit installation was successful, do the following:

1. View Client services project types.
   a. Click **File** → **New** → **Other** to confirm that the *Client Services* project type is available from the menus.
   b. Expand **Client Services**. You should see a dialog like Figure 10-6 on page 269 listing the Client Services wizards. Click **Cancel** when done.


New

Select a wizard

Create a Dynamic Web project

Wizards:

- Java Project
- Web Service
- Client Services
  - Client Services Embedded Transaction Project
  - Client Services Fragment Project
  - Client Services Project
  - Client Services Web Project
  - Convert Project to Client Services Project
  - Migrate Extension Services Project
  - Platform Builder Project
  - Mobile Web Services
    - Mobile Web Services Client
    - Mobile Web Services Provider
    - Mobile Web Services Provider Security
- Crystal Report

☐ Show All Wizards.

Figure 10-6 Client Services wizards

2. Verify the WebSphere Everyplace Client Toolkit appears in the product configuration of Rational Application Developer.

   a. Click **Help** → **Software Updates** → **Manage Configuration**.

   b. When the Product Configuration dialog appears, verify that IBM WebSphere Everyplace Client Toolkit 6.0.0 is installed in the list. Click **Cancel** when done.

Note: By default, the toolkit is installed in the <rsa_home>/sdpisv/eclipse directory (for example, C:\Program Files\IBM\Rational\SDP\6.0\sdpisv\eclipse). You can also click the Installation History icon to view update activities from the product installation.

Configure WebSphere Everyplace Client Toolkit

If a new work space is created within Rational Software Architect, WebSphere Everyplace Client Toolkit will detect this (15 second intervals) and prompt the user with a dialog to allow registering bundles with the work space. If the user declines this update, they will need to manually configure the WebSphere Everyplace Client Toolkit for the work space.

To configure WebSphere Everyplace Client Toolkit, do the following:

1. Ensure after the WebSphere Everyplace Client Toolkit installation, that you click **OK** to set the preferences.
2. Select **Window** → **Preferences**.
3. Expand **Java** and click **Installed JREs**.
4. Check the check box next to "WebSphere Everyplace Deployment v6 JRE."
5. Expand **Plug-in Development** → **Target platform**.
6. Select **<rational_dir>\sdpisv\eclipse\plugins\com.ibm.pvc.wct.runtimes.6.0.0.<*date*>\eclipse** (default) from the Location drop-down list.
7. Click **Apply**.
8. Click **OK**.

10.5 Import the completed sample code

There are several approaches for demonstrating how to develop an application sample. We have listed a few possibilities:

- Learn from completed sample.

  A developer may learn from a completed sample. In this case, simply import the ITSO_MobileAdjuster_PI.zip into the work space.

  Note: This is the approach described in this section.

- Import completed sample and create new projects and code.

  Another approach is to import the completed work, refactor selected client-side projects, and then create the client-side application projects using the WebSphere Everyplace Client Toolkit.

  When using this approach, the developer will import the server-side projects, manually create client-side projects using the WebSphere Everyplace Client Toolkit provided functionality, create the packages, and then either import the complete Java source files or create them from scratch. In our example, we choose to import the complete Java source and explain the key parts of this source code.

Note: This approach is described in Chapter 11, “Application development” on page 279.

- Create all samples from scratch.

  This approach is beyond the scope of this book.

This section describes how to import the completed ITSO Mobile Adjuster example project interchange file (ITSO_MobileAdjusterPI.zip) for the server-side and client-side application components into Rational Software Architect.

Complete the following steps to prepare the environment:

1. Download sample application code.
2. Create a new workspace.
3. Import the application project interchange file.
4. Create a new server.
5. Clean build.

10.5.1 Download sample application code

Appendix E, “Additional material” on page 503, describes how to download and unpack the ITSO sample code for this book.

After unpacking the ITSO sample code 6775code.zip, you will have a C:\6775code directory containing the ITSO MobileAdjuster sample source files and deployable artifacts.

10.5.2 Create a new workspace

While writing this book, we found that it was necessary to restart Rational Software Architect (or Rational Application Developer) after creating a new workspace. After creating a new workspace, the WebSphere Everyplace Client Toolkit detects the workspace and prompts you to add the bundles. After the bundles have been registered, we recommend that you restart Rational Software Architect to properly pick up the registered bundles.

1. Launch Rational Software Architect and create a new workspace (for example, c:\workspace).
2. Close the Welcome screen.
3. When prompted with `Do you want to automatically set the Preferences for the WebSphere Everyplace Client Toolkit?`, click **OK**.

Note: The WebSphere Everyplace Client Toolkit V6.0 will detect the new workspace and register the bundles.

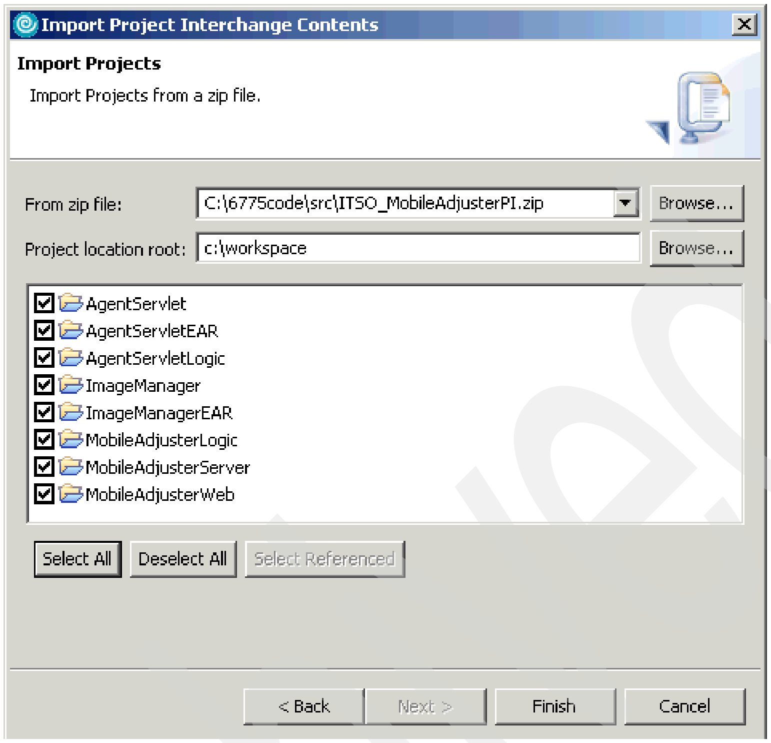
4. Restart the IBM Rational Software Architect for the WebSphere Everyplace Client Toolkit to work properly.

10.5.3 Import the application project interchange file

Rational Application Developer provides a very portable project packaging file named a project interchange file. We have packaged the ITSO MobileAdjuster application source code and projects as a project interchange file.

To import the ITSO_MobileAdjusterPI.zip project interchange file into Rational Software Architect, do the following:

1. Start Rational Software Architect.
2. Open the J2EE perspective Project Explorer view.
3. Select **File** → **Import**.
4. When the Import Wizard appears, select **Project Interchange** and then click **Next**.
5. In the Import projects dialog, browse to the C:\6775code\src folder and select **ITSO_MobileAdjusterPI**.zip. Click **Open**.
6. Click **Select All** (as seen in Figure 10-7 on page 273) and then click **Finish**.


Import Project Interchange Contents

Import Projects

Import Projects from a zip file.

From zip file: C:\6775code\src\ITSO_MobileAdjusterPI.zip

Project location root: c:\workspace

- ☑ AgentServlet
- ☑ AgentServletEAR
- ☑ AgentServletLogic
- ☑ ImageManager
- ☑ ImageManagerEAR
- ☑ MobileAdjusterLogic
- ☑ MobileAdjusterServer
- ☑ MobileAdjusterWeb

Figure 10-7   ITSO MobileAdjuster - Import project interchange file

10.5.4 Create a new server

To create a new WebSphere Everyplace Deployment server that we can add the MobileAdjusterWeb project to, do the following:

Note: We found that this step helped resolve meta data problems with the project interchange file.

1. Open the J2EE perspective.
2. Right-click in the Servers view and select **New** → **Server**.
3. Select **WebSphere Everyplace Deployment v6.0** and click **Next**.
4. When the WebSphere Everyplace Deployment Client Server dialog appears, we accepted the default settings and clicked **Next**.
5. Select **MobileAdjusterWeb**, click **Add >** to add to the Configured products list, and then click **Finish**.

10.5.5 Clean build

We recommend that you perform a clean build within your workspace. When done, you should not have errors in the Problems view. You may still see some warnings.

10.6 Create client configuration and run application

This section describes how to run the ITSO MobileAdjuster server-side application on the remote Access Server node (WebSphere Everyplace Deployment Server) and the client-side application in the WED client from within Rational Software Architect.

10.6.1 Create a unique user ID for development

When using DB2 Everyplace, it is important to understand that a unique user ID is required for each device used by DB2 Everyplace. For this reason, we created the *devadjuster1* user ID to be used when running the MobileAdjuster application and running within the WebSphere Everyplace Client Toolkit Client runtime of Rational Software Architect.

For details on creating the devadjuster1 user ID and adding the user to the DB2eAdjusters group, refer to 13.2.2, “Create users and groups” on page 350.

10.6.2 Deploy the server-side application

The WebSphere Everyplace Deployment product does not include a test environment that can be installed within the Rational Software Architect as a test environment. For this reason, server-side application components are developed within Rational Software Architect, but need to be deployed to a remote WebSphere Everyplace Deployment Server to fully test end-to-end applications.

For details on deploying the server-side application components for the ITSO MobileAdjuster example, refer to 13.2, “Server-side application deployment” on page 348.

After deploying the server-side application, ensure that the WebSphere Everyplace Deployment servers are started.

10.6.3 Create the client configuration

To create the WebSphere Everyplace Deployment Client configuration to run the client-side application within Rational Software Architect, do the following:

1. Perform a clean build.

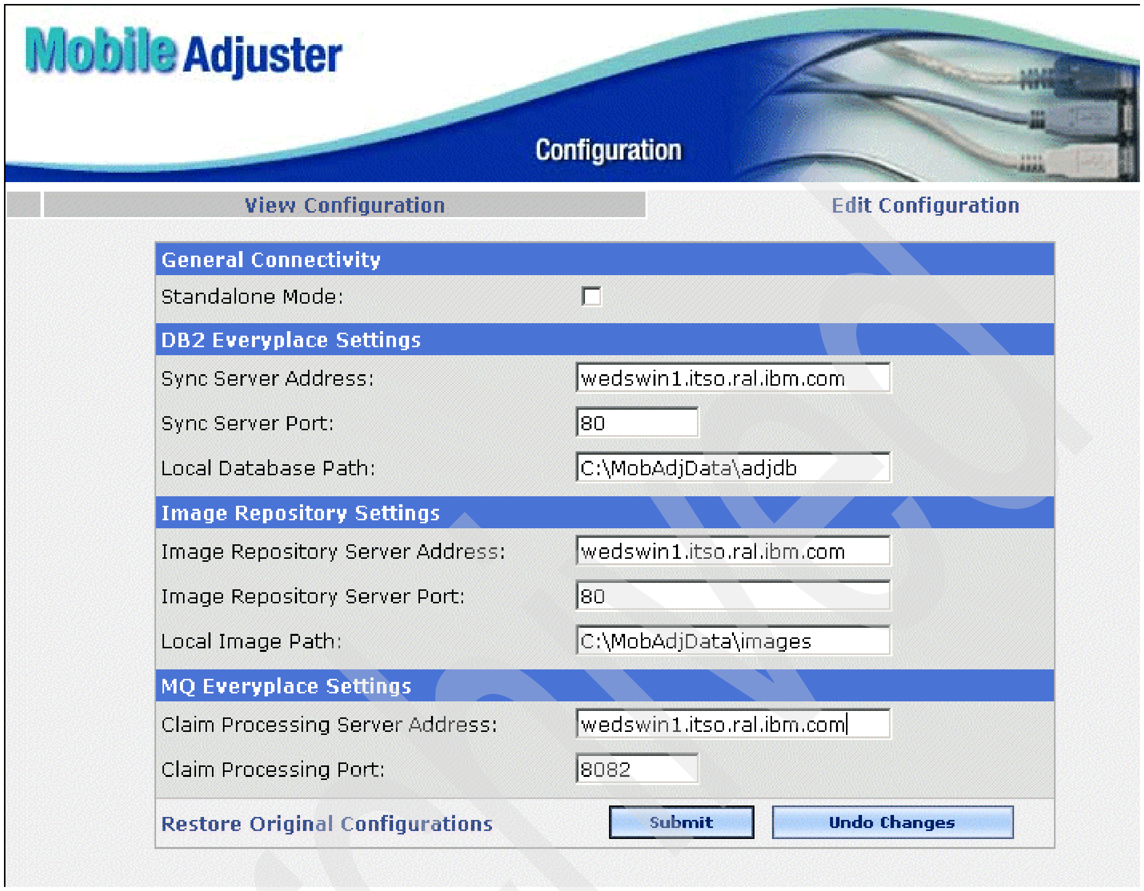
   Ensure that the MobileAdjusterWeb and MobileAdjusterLogic projects have no compilation errors.

2. Select **Run** → **Run** from the main menu.
3. Select **WebSphere Everyplace Deployment** from the list of configurations, and click **New**.
4. When the Create, manage and run configurations dialog appears, do the following:
   a. Enter `WED Client configuration` in the Name field.
   b. Click the **Plug-ins** tab.
   c. Expand **Workspace Plug-ins**, and verify that MobileAdjusterWeb and MobileAdjusterLogic are checked.
   d. Click **Apply**.

10.6.4 Run the client configuration

To run the client-side MobileAdjuster application on the WED Client configuration within Rational Application Developer, do the following:

1. Select **Run** → **Run** from the main menu.
2. Select the **WED Client configuration** and click **Run**.
3. Click **Application** → **Open** from the WebSphere Everyplace Deployment main menu, and verify that the following three applications are listed:
   - MobileAdjuster
   - MobileAdjuster Config
   - MobileAdjuster Images
4. Configure the MobileAdjuster application for the proper settings of the WebSphere Everyplace Deployment Server.
   a. Click **Open** → **Mobile Adjuster Config**.
   b. When the Mobile Adjuster Configuration page appears, click **Edit Configuration**.
   c. Enter your configuration preferences. For the ITSO example runtime environment, we entered the values displayed in Figure 10-8 on page 276, and then clicked **Submit**.



General Connectivity	
Standalone Mode:	☐
DB2 Everyplace Settings	
Sync Server Address:	wedswin1.itso.ral.ibm.com
Sync Server Port:	80
Local Database Path:	C:\MobAdjData\adjdb
Image Repository Settings	
Image Repository Server Address:	wedswin1.itso.ral.ibm.com
Image Repository Server Port:	80
Local Image Path:	C:\MobAdjData\images
MQ Everyplace Settings	
Claim Processing Server Address:	wedswin1.itso.ral.ibm.com
Claim Processing Port:	8082

Figure 10-8 Mobile Adjuster - Configuration settings

5. To verify the MobileAdjuster application, do the following:

   a. Click **Open** → **Mobile Adjuster**.

   b. When the Mobile Adjuster Log in page appears, enter the adjuster user ID and password (for example, `devadjuster1`), and then click **Login**.

> **Restriction:** At the time of writing, a unique user ID was required for each device used by DB2 Everyplace. In our example, we have a WED Client node in the runtime and a WECT Client in the development environment both sharing the same WED Server. The user ID that we use to log in to the MobileAdjuster application must be unique; thus, we created the devadjuster1 user ID to be used while testing in the development environment.

c. The first time you log in there will be no local claims or customer data on the client. The Store data will display zeros for policies, claims, and damages until you refresh the data.

Note: Prior to synchronizing the data to the client, the database tables will be empty. As a result, you will see the following error message, which can safely be ignored:

```
[AdjusterDatabase] Error counting table CLAIMS: java.sql.SQLException
...
```

d. Click **Refresh Data**.

After the refresh is complete, you should see the Stored data values change.

Note: If the Refresh Data (sync) fails, check the following:

- Verify the network configuration.
- Ensure you can authenticate (DB2e Client → Server).

For a full application walkthrough, refer to Chapter 15, “Application walkthrough” on page 407.

11 Application development

In Chapter 10, “Development environment installation” on page 255, we learned how to implement the development environment used to develop applications representative of the Distributed Rich Client application pattern. The core components of the development environment include IBM Rational Development tooling (Software Architect or Application Developer), plus the extended capability provided by the IBM WebSphere Everyplace Client Toolkit.

Our focus in this chapter is to demonstrate how to develop applications representative of the Distributed Rich Client application pattern to be deployed on WebSphere Everyplace Deployment Client for Windows and Linux (client) and WebSphere Everyplace Deployment (server).

The chapter is organized into the following sections:

- Prepare for application development.
- Develop client-side application logic.
  - Develop disconnectable app using transactional messaging.
  - Develop disconnectable app using relational database sync.
  - Develop disconnectable app using Mobile Web Services.
- Develop disconnectable app with local Web UI.
- Package the application for deployment.

11.1 Prepare for application development

This section describes the steps necessary to prepare for developing the ITSO MobileAdjuster client-side application representative of the Distributed Rich Client application pattern.

11.1.1 Development environment installation

For details refer to Chapter 10, “Development environment installation” on page 255.

11.1.2 Download ITSO sample code

Appendix E, “Additional material” on page 503, describes how to download and unpack the ITSO sample code for this book.

After unpacking the ITSO sample code 6775code.zip, you will have a C:\6775code directory containing the ITSO MobileAdjuster sample source files and deployable artifacts.

Unpack the C:\6775code\src\ITSO_clientsrc.zip to the c:\temp directory. We import some of the Java source files from this directory structure.

11.1.3 Create a new workspace

While writing this book we found that it was necessary to restart Rational Software Architect (or Rational Application Developer) after creating a new workspace. After creating a new workspace, the WebSphere Everyplace Client Toolkit will detect the workspace and prompt you to add the bundles. After the bundles have been registered, we recommend that you restart Rational Software Architect to properly pick up the registered bundles.

1. Launch Rational Software Architect and create a new workspace (for example, c:\workspace).
2. Close the Welcome screen.
3. When prompted with `Do you want to automatically set the Preferences for the WebSphere Everyplace Client Toolkit?`, click **OK**.

Note: The WebSphere Everyplace Client Toolkit V6.0 detects the new workspace and registers the bundles.

4. Restart the IBM Rational Software Architect for the WebSphere Everyplace Client Toolkit to work properly.

11.1.4 Import source code from ITSO_MobileAdjuster.zip

There are a several ways to go about demonstrating how to develop an application sample. We have listed a few possibilities:

- Learn from completed sample.

  A developer may learn from a completed sample. In this case, simply import the ITSO_MobileAdjuster_PI.zip into the workspace.

  Note: If you choose this method, you will only reference the remaining sections in the chapter explaining the code.

  This method is described in detail in 10.5, "Import the completed sample code" on page 270.

- Import completed sample and create new projects and code.

  Another approach is to import the completed work, refactor selected client-side projects, and then create the client-side application projects using the WebSphere Everyplace Client Toolkit.

  When using this approach, the developer will import the server-side projects, manually create client-side projects using the WebSphere Everyplace Client Toolkit provided functionality, create the packages, and then either import the complete Java source files or create them from scratch. In our example, we choose to import the complete Java source and explain the key parts of this source code.

  Note: The remaining sections in this chapter focus on this approach.

- Create all samples from scratch.

  This approach is beyond the scope of this book.

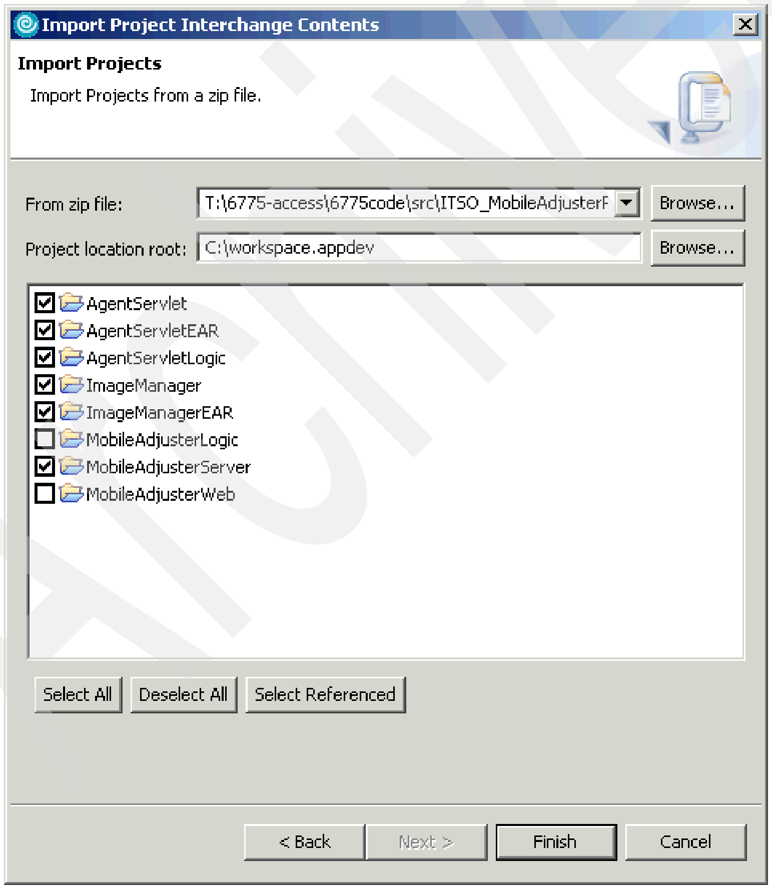
The Rational Developer products provide a very portable project packaging file named a project interchange file. We have packaged the ITSO MobileAdjuster application source code and projects as a project interchange file (ITSO_MobileAdjusterPI.zip).

To import the ITSO_MobileAdjusterPI.zip project interchange file into Rational Software Architect, do the following:

1. Start Rational Software Architect.
2. Open the J2EE perspective Project Explorer view.
   a. Select **Window** → **Open Perspective** → **Other**.

   b. Select **J2EE** and click **OK**.
   c. Select **Window** → **Show View** → **Project Explorer**.
3. Select **File** → **Import**.
4. When the Import Wizard appears, select **Project Interchange** and then click **Next**.
5. In the Import projects dialog, browse to the C:\6775code\src folder and select **ITSO_MobileAdjusterPI.zip**. Click **Open**.
6. Click **Select All** (as seen in Figure 11-1) and then click **Finish**.

> **Note:** You will see errors after the import. Do not be alarmed. By continuing in the documented procedure these errors will be resolved.



From zip file: T:\6775-access\6775code\src\ITSO_MobileAdjusterF

Project location root: C:\workspace.appdev

- [x] AgentServlet
- [x] AgentServletEAR
- [x] AgentServletLogic
- [x] ImageManager
- [x] ImageManagerEAR
- [ ] MobileAdjusterLogic
- [x] MobileAdjusterServer
- [ ] MobileAdjusterWeb

Figure 11-1 ITSO MobileAdjuster - Import project interchange file

Note: After importing the projects, it will take several minutes to build (assuming Automatic Build is turned on). You may see many errors and warnings. We will resolve these in later steps.

11.2 Develop client-side application logic

This section describes how to perform the following tasks to create the client-side MobileAdjuster logic:

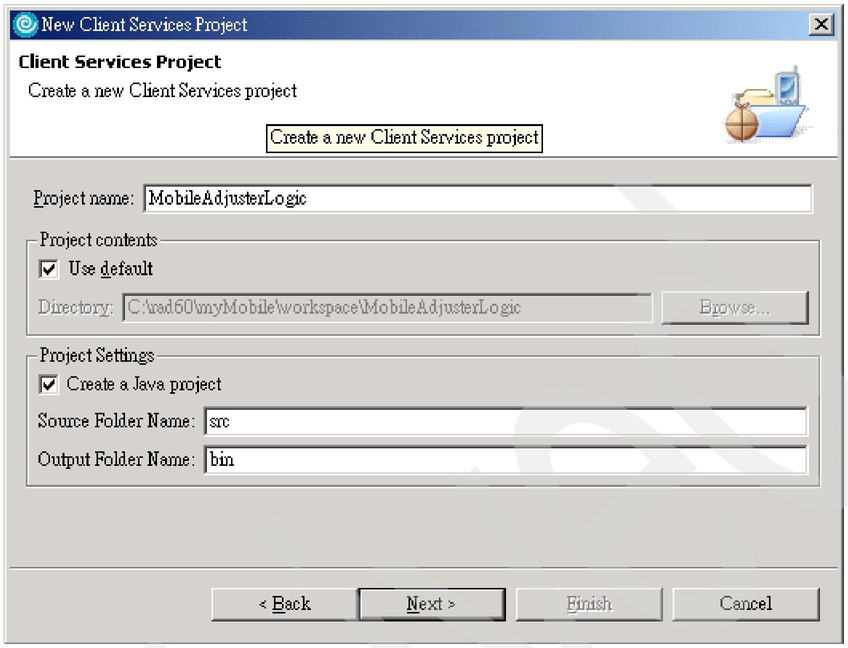
- Create a new Client Services Project (logic).
- Import sample Java source into Client Services Project.
- Develop disconnectable app using transactional messaging.
- Develop disconnectable app using relational database sync.
- Develop disconnectable app using Mobile Web Services.
- Edit the MANIFEST.MF for exporting packages.

11.2.1 Create a new Client Services Project (logic)

This section describes how to create a Client Services Project, which will be used to develop the business application logic (DB2e, MQe, Web Services).

To create the Client Services project, do the following:

1. To launch a new Client Services Project wizard, select **File** → **New** → **Project**.
2. Expand **Client Services**, select **Client Services Project**, and then click **Next**.
3. When the Client Services Project dialog appears, enter `MobileAdjusterLogic` in the Project name field, as seen in Figure 11-2, and then click **Next**.


New Client Services Project

Client Services Project

Create a new Client Services project

Project name: MobileAdjusterLogic

Project contents

☑ Use default

Directory: C:\rad60\myMobile\workspace\MobileAdjusterLogic

Project Settings

☑ Create a Java project

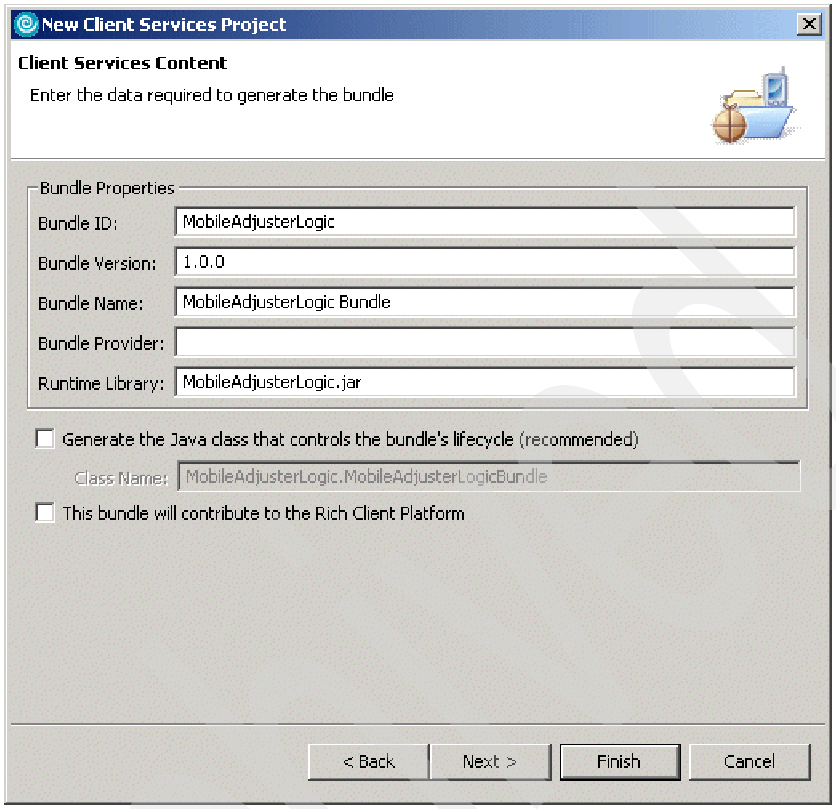
Source Folder Name: src

Output Folder Name: bin

Figure 11-2 New Client Services Project wizard

4. When the Client Services Content dialog appears, uncheck “Generate the Java class that controls the bundle’s lifecycle” and “This bundle will contribute to the Rich Client Platform,” as shown in Figure 11-3, and then click **Next**.

> **Note:** In our example we do not use the this class.


New Client Services Project

Client Services Content

Enter the data required to generate the bundle

Bundle Properties

Bundle ID: MobileAdjusterLogic

Bundle Version: 1.0.0

Bundle Name: MobileAdjusterLogic Bundle

Bundle Provider:

Runtime Library: MobileAdjusterLogic.jar

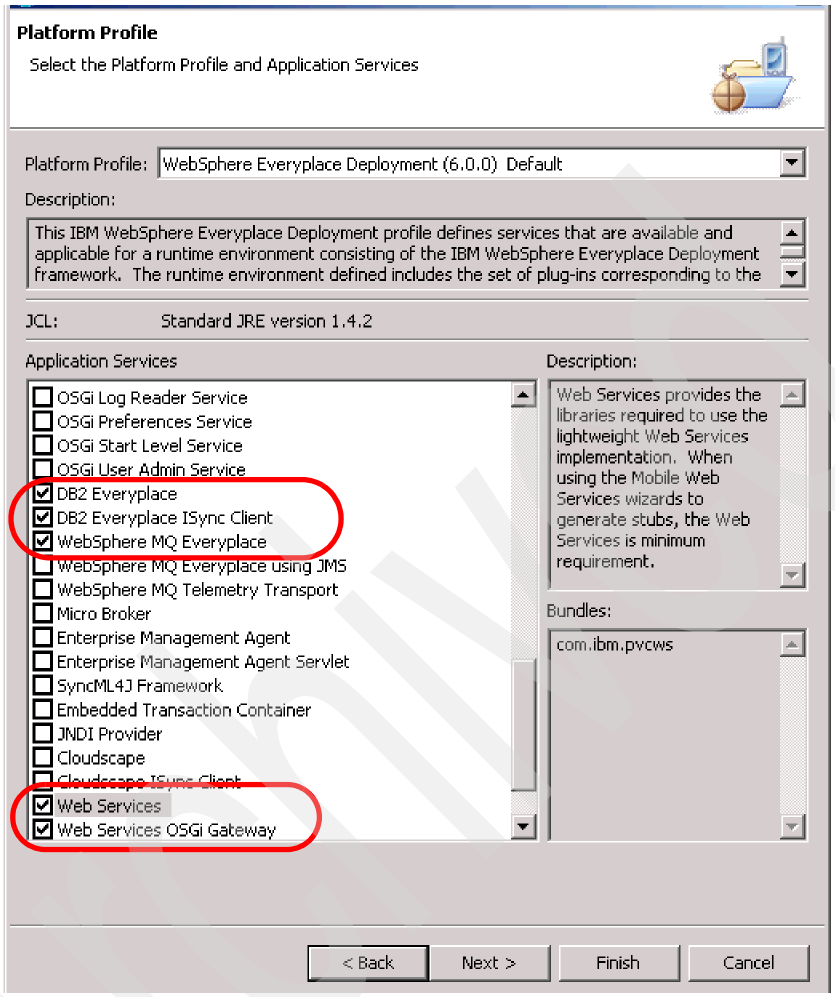
☐ Generate the Java class that controls the bundle's lifecycle (recommended)

Class Name: MobileAdjusterLogic.MobileAdjusterLogicBundle

☐ This bundle will contribute to the Rich Client Platform

Figure 11-3 Client Services Content

5. When the Platform Profile dialog appears, do the following:
   a. Select **WebSphere Everyplace Deployment (WebSphere Everyplace Deployment(6.0.0) Default** from the Platform Profile drop-down list.
   b. Check the following Application services needed to develop the ITSO MobileAdjuster business logic:
      - Check **DB2 Everyplace**.
      - Check **DB2 Everyplace ISync Client**.
      - Check **WebSphere MQ Everyplace**.
      - Check **Web Services**.
      - Check **Web Services OSGi Gateway**.
   c. When done the dialog should look like Figure 11-4. Click **Next**.



Platform Profile

Select the Platform Profile and Application Services

Platform Profile: WebSphere Everyplace Deployment (6.0.0) Default

Description:

This IBM WebSphere Everyplace Deployment profile defines services that are available and applicable for a runtime environment consisting of the IBM WebSphere Everyplace Deployment framework. The runtime environment defined includes the set of plug-ins corresponding to the

JCL: Standard JRE version 1.4.2

Application Services

- [ ] OSGi Log Reader Service
- [ ] OSGi Preferences Service
- [ ] OSGi Start Level Service
- [ ] OSGi User Admin Service
- [✓] DB2 Everyplace
- [✓] DB2 Everyplace ISync Client
- [✓] WebSphere MQ Everyplace
- [ ] WebSphere MQ Everyplace using JMS
- [ ] WebSphere MQ Telemetry Transport
- [ ] Micro Broker
- [ ] Enterprise Management Agent
- [ ] Enterprise Management Agent Servlet
- [ ] SyncML4J Framework
- [ ] Embedded Transaction Container
- [ ] JNDI Provider
- [ ] Cloudscape
- [ ] Cloudscape ISync Client
- [✓] Web Services
- [✓] Web Services OSGi Gateway

Description:

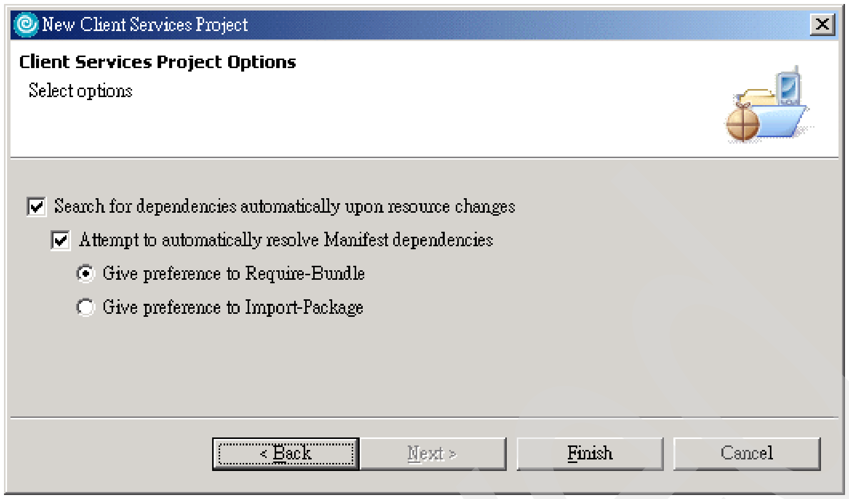
Web Services provides the libraries required to use the lightweight Web Services implementation. When using the Mobile Web Services wizards to generate stubs, the Web Services is minimum requirement.

Bundles:

com.ibm.pvcws

Figure 11-4 Platform Profile dialog - Select Application Services used in the project

6. When the Client Services Project Options dialog appears, we accepted the default values, as seen in Figure 11-5 on page 287, and then clicked **Finish**.


New Client Services Project

Client Services Project Options

Select options

[✓] Search for dependencies automatically upon resource changes

[✓] Attempt to automatically resolve Manifest dependencies

[●] Give preference to Require-Bundle

( ) Give preference to Import-Package

Figure 11-5 select Client Services Project Options

11.2.2 Import sample Java source into Client Services Project

This section describes how to import the Java source files and create the corresponding packages in the MobileAdjusterLogic Client Services project.

The MobileAdjusterLogic project requires the packages listed in Table 11-1.

Table 11-1 ITSO MobileAdjusterLogic project packages

Package
com.ibm.mobile.adjuster.beans
com.ibm.mobile.adjuster.db2e
com.ibm.mobile.adjuster.mqe
com.ibm.mobile.adjuster.service
com.ibm.mobile.images
com.ibm.mobile.server.messages.base
com.ibm.mobile.server.messages.base.impl
com.ibm.mobile.server.transport
com.ibm.mobile.server.transport.mqe

In our example, we have provided the completed source files in the proper packages listed in Table 11-1 on page 287. To import the Java source and create the packages, do the following:

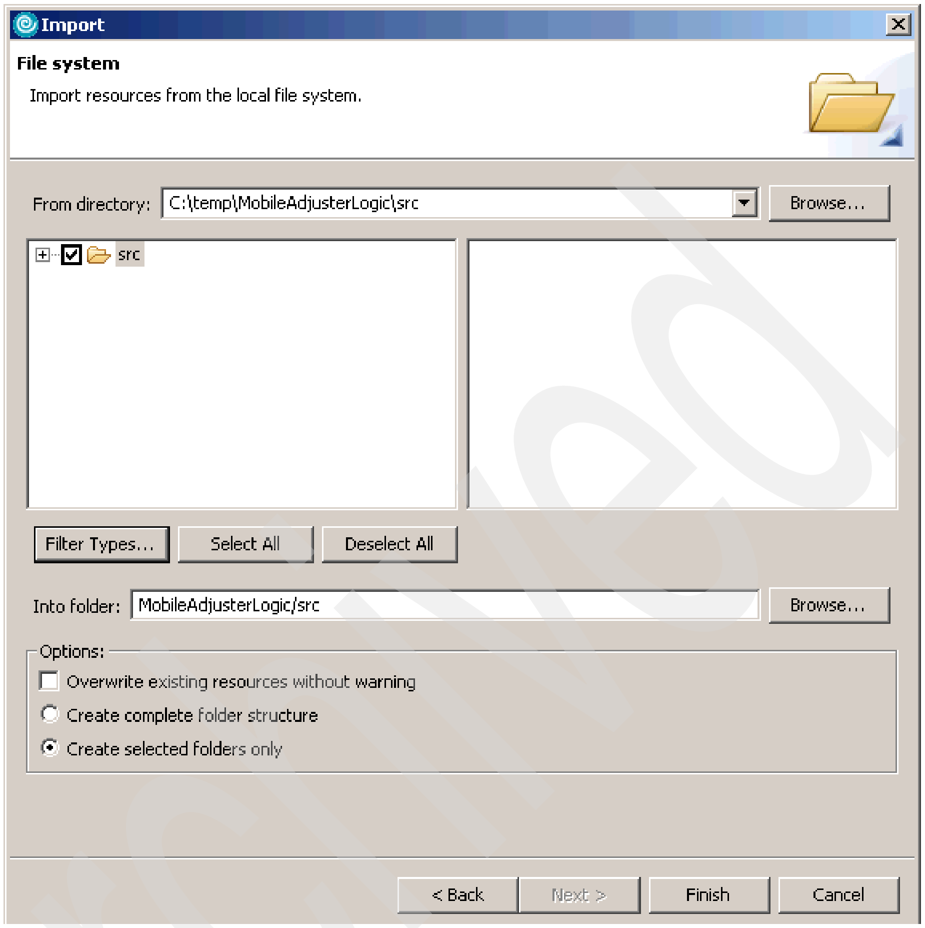
1. Open the Java perspective.
2. Expand **MobileAdjusterLogic**, right-click the **src** folder, and select **Import**.
3. When the Import dialog appears, select **File system** and click **Next**.
4. When the File system dialog appears, do the following:
   a. Enter `c:\temp\MobileAdjusterLogic\src` in the From directory.

      This directory was created as a result of unzipping the ITSO_clientsrc.zip in 11.1.2, "Download ITSO sample code" on page 280.

   b. Check **src**.

   c. Click **Filter types**. Check **.java** files and click **OK**.

   d. When done, the dialog should look like Figure 11-6 on page 289. Click **Finish**.


Import

File system

Import resources from the local file system.

From directory: C:\temp\MobileAdjusterLogic\src

[✓] src

Into folder: MobileAdjusterLogic/src

Options:

[ ] Overwrite existing resources without warning

( ) Create complete folder structure

(●) Create selected folders only

Figure 11-6 Import Java class files

5. Table 11-2 lists the class names and corresponding packages listed previously in Table 11-1 on page 287. The AdjusterService is the controller for the MobileAdjusterLogic.

Table 11-2 MobileAdjusterLogic class names and packages

Class name	Package
AdjusterService	com.ibm.mobile.adjuster.service
MobileConstants	com.ibm.mobile.adjuster.service
ClaimInfo	com.ibm.mobile.adjuster.beans
CustomerInfo	com.ibm.mobile.adjuster.beans

Class name	Package
MessageInfo	com.ibm.mobile.adjuster.beans
PartInfo	com.ibm.mobile.adjuster.beans

Tip: Creating from scratch

If you were creating the packages and Java class files from scratch you would do the following:

1. Create package.
   a. Expand the **MobileAdjusterLogic** project, right-click **src**, and select **New** → **Packages**.
   b. Enter the package name (for example, com.ibm.mobile.adjuster.beans) and click **Finish**.
2. Create the Java class file.
   a. Right-click the package and select **New** → **Class**.
   b. Enter the class name and click **Finish**.

11.2.3 Develop disconnectable app using transactional messaging

The basic client functions of MQe are represented in the classes listed in Table 11-3. These classes contain functions and APIs that are more comprehensive than necessary for our Mobile Adjuster application. However, they are provided as examples to developers who are looking to implement an MQe solution on the client with little overhead.

Note: For more details on the functionality and capabilities contained within these MQe samples, refer to the following IBM WebSphere MQ Everyplace product guides:

- *Application Programming Guide, IBM WebSphere MQ Everyplace*, SC34-6278-01
- *Configuration Guide, IBM WebSphere Everyplace MQ*, SC34-6283-02

Table 11-3 Classes that help implement MQe on the client

Class name	Package
ClaimMsgConstants	com.ibm.mobile.adjuster.mqe

Class name	Package
ClaimTransportListener	com.ibm.mobile.adjuster.mqe
ClaimTransport	com.ibm.mobile.adjuster.mqe
MQeClaimMsgObject	com.ibm.mobile.adjuster.mqe
MQeQM	com.ibm.mobile.adjuster.mqe
IMessage	com.ibm.mobile.server.messages.base
IMessageEnvelope	com.ibm.mobile.server.messages.base
BaseMessageEnvelope	com.ibm.mobile.server.messages.base.impl
ITransport	com.ibm.mobile.server.messages.transport
ITransportListener	com.ibm.mobile.server.messages.transport
BaseMQeTransport	com.ibm.mobile.server.messages.transport.mqe

The following services need to be implemented in code using the MQe APIs:

- Submit a claim to the server.

  When submitting a claim to the WED Server, the claim information is inserted into the local client database, then wrapped as an MQe message and placed in the local message queue. This message is delivered to the WED server as an MQe message when the client is connected to the network.

- Check for new messages.

  The user manually checks the client message queue for new messages by clicking the **Check Message queue** link. This triggers communication with the MQe Server, and the MQe Server sends a message back to the MQe client if a message exists. If a new message is detected, it will be processed by the client, and any information present in the message is updated in the client's local database. Once stored in the database, this information is available for viewing by the user.

Submit a claim to the server

The newly created claim is submitted through the submitClaim() method in the controller class AdjusterService.

The submitClaim method performs the following actions (see Example 11-1 on page 292 for code details):

1. Determines whether the client is disconnected from the network (for example, working in standalone mode).

2. Inserts or modifies claim information in the local client database.
3. Creates new or calls existing ClientTransportListener and BaseMQeTransport objects. This initializes the local message queue and queue manager for the upcoming message transmission. The BaseMQeTransport class creates the MQeQM queue manager as well, if it has not already been created. (Please read the note below regarding this step.)

Note: The process encompasses many actions beyond the scope of this book, but that are outlined in the MQe documentation. However, we would like to highlight one important characteristic of the MQe Queue Manager on the client. Currently, only one Queue Manager per JVM can reside on a particular system. This means that multiple applications running within the WebSphere Everyplace Deployment Client for Windows and Linux and utilizing MQe must use the same Queue Manager.

Present in the Mobile Adjuster sample code is an MQeQM class that checks to see if an existing MQe Queue Manager exists within the running JVM and, if so, allows the requesting application to use this Queue Manager. If the Queue Manager does not exist, it creates a new one with a unique timestamp. The MQeQM object is instantiated in the BaseMQeTransport class. It is highly recommended to any developer using MQe in production-level client applications to implement a similar system of checking.

Please refer to the code in MQeQM.java in the Mobile Adjuster sample code (too large to include in this book).

4. Prepare the claim information to be sent as an MQe message.
5. Send the message to the message queue for transmission.

Example 11-1 submitClaim() method from AdjusterService.java

```
public void submitClaim(ClaimInfo ci/*, String userID*/) {
      log("Submitting Claim "+ci.getClaimID());

      //If standalone, change the status to Processing, to signify no MQe message will be sent
      //Otherwise, change to Submitted
      if (isStandalone())
         ci.setStatus(ClaimInfo.PROCESSING);
      else
         ci.setStatus(ClaimInfo.SUBMITTED);

      //If the database call to insert the claim fails, the claim probably exists already in
the database
```

```
      boolean success = getDatabase().insertClaim(ci);
      if (!success)
         success = getDatabase().modifyClaim(ci);

      //No MQe message is sent if the database updates failed or if we are in standalone mode
      if (success && !isStandalone()) {
         BaseMQeTransport transport = getListener().getTransport();

         //Confirm that the MQe transport was setup successfully
         if (transport != null) {
            try {
               //log("Sending claim message");
               MQeClaimMsgObject msgObj = new MQeClaimMsgObject();
               msgObj.setMessageReasonCode(ClaimMsgConstants.CLAIM_REQ_CODE);
               msgObj.setMessageID("CID001"); //$NON-NLS-1$
               msgObj.storeClaim(ci);
               msgObj.setReplyToQueueManagerName(transport.getDefaultQueueManagerName() );
               msgObj.setReplyToQueueName(transport.getReplyQueueName());

               log("Submitting Claim: " + msgObj.getDisplayString());
               IMessageEnvelope env =
                  new BaseMessageEnvelope(
                     transport.getServerQueueManagerName(),
                     transport.getRequestQueueName(),
                     msgObj);
               getListener(/*userID*/).getTransport().send(env);
               log("Claim message sent to queue!");
            } catch (Exception e) {
               e.printStackTrace();
            }
         } else { // if the transport setup returned a null transport
            log(" Unable to submit claim to Claim Processing Server, due to MQe transport
setup failure");
         }
      }
   }
```

Check for new messages

A manual check for new messages is performed by calling the checkMessageQueue() method also located in the AdjusterService controller class (Example 11-2).

Example 11-2 checkMessageQueue() method from AdjusterService.java

```
   public void checkMessageQueue() {
      if (isStandalone())
         return;
      try {
         BaseMQeTransport transport = getListener().getTransport();
         if (transport != null) {
            log("Checking Message Queue");
            ((ClientTransport) transport).triggerTransmission();
         }
      } catch (Exception e) {
         logError("Exception checking message queue: +e");
      }
      return;
   }
```

This method primarily consists of calling the triggerTransmission() MQe API. This API initiates communication between the MQe server and MQe client, and performs several underlying proprietary message delivery actions. If a new message is delivered to the MQe client and found in the message queue, the content is extracted and processed accordingly.

Note: The MobileAdjusterServer project includes the MQe server-side application components that the MQe client code interfaces with. The MobileAdjusterServer project (MQe server) is beyond the scope of this book.

11.2.4 Develop disconnectable app using relational database sync

This section describes how to develop a disconnectable application using DB2 Everyplace (DB2e) JDBC to access the MobileAdjuster database.

When you are developing code leveraging the DB2 Everyplace ISync APIs, be sure to select the DB2 Everyplace ISync client when creating the Client Service Project (see Figure 11-4 on page 286).

Note: For more information refer to *Application and Development Guide, IBM DB2 Everyplace V8.2*, SC18-7185-04, product guide.

We implemented the classes listed in Table 11-4.

Table 11-4 Classes that implement data access and sync

Class name	Package
AdjusterDatabase	com.ibm.mobile.adjuster.db2e
DatabaseSyncManager	com.ibm.mobile.adjuster.db2e

Connect to DB2e database

The JDBC application code can be found in AdjusterDatabase.java. The first step to implement a JDBC application is to get the connection to the database. The application uses the DriverManager class to obtain a connection. The URL String that is passed in to the getConnection method for DB2e is shown in Example 11-3.

Example 11-3 Sample snippet to connect DB2e database

```
private String adjdbUrl = null;
adjdbUrl = "jdbc:db2e:" + dbDir.getAbsolutePath() + File.separator;

   private Connection getConnection() throws Exception {
      Connection con = DriverManager.getConnection(adjdbUrl);
      return con;
   }
```

Once the JDBC driver is loaded and a connection to the database is established, we are ready to perform operations on the database. Example 11-4 includes a basic SQL statement that retrieves all customers from the CUSTOMERS table.

Retrieve customer data from local DB2e database

The major steps to retrieve customer data from the local database are as follows:

1. Connect to the database. The application uses the DriverManager class to obtain a connection.
2. Create a Statement object using the connection obtained from the DriverManager.
3. Select all records from the CUSTOMERS table; the result set from the query is returned as a ResultSet Object. Then retrieve the rows by calling the next() method of the java.sql.ResultSet interface.
4. Finally, JDBC objects must be closed to release the resources.

In Example 11-4, we demonstrate a snippet AdjusterDatabase source code that contains comments to display where the steps explained above are used.

Example 11-4 Sample snippet AdjusterDatabase.java to query customers

```
   public Vector getCustomers() {
      //log("getting customers");

      Vector customers = new Vector();
      Connection con = null;
      Statement st = null;
      ResultSet rs = null;
      try {
         con = getConnection(); //step 1
         st = con.createStatement(); //step 2
         rs = st.executeQuery("SELECT CUSTOMERID, NAME, DEDUCTIBLE, EXPIRES
FROM CUSTOMERS ORDER BY NAME ASC"); //step 3

         while (rs.next()) {
            CustomerInfo ci = new CustomerInfo();
            ci.setCustomerID(rs.getString("CUSTOMERID"));
            ci.setName(rs.getString("NAME"));
            ci.setDeductible(rs.getInt("DEDUCTIBLE"));
            ci.setPolicyExpires(rs.getDate("EXPIRES"));
            customers.addElement(ci);
         }
         log("Found " + customers.size() + " customers");
      } catch (Exception e) {
         logError("Error getting customers: "+e);
         //e.printStackTrace();
      } finally { //step 4
         close(rs);
         close(st);
         close(con);
      }
      return customers;
   }
```

Update claim status in local DB2e database

In Example 11-5, we demonstrate another snippet of code that is used to update the local database content by using DB2e. The updateClaimStatus method was called by the updateClaimStatus method in the AdjusterService module.

The major steps to update the claim status for the local DB2e database are as follows:

1. Connect to the database. The application uses the DriverManager class to obtain a connection.
2. Create a Statement object.
3. Update the claim status of the database record selected. This is performed using the executeUpdate(String sql) method of the java.sql.Statement interface.
4. Release JDBC resources.

Example 11-5 Sample snippet AdjusterDatabase.java to update a claim status

```
   public boolean updateClaimStatus(String cid, String status) {
      log("*** Updating claim "+cid+" status to "+status);
      if (isDummy)
         return false;

      boolean success = false;
      Connection con = null;
      Statement st = null;

      try {
         con = getConnection(); //step 1
         st = con.createStatement(); //step 2
         String updateSQL = "UPDATE CLAIMS SET STATUS='"+status+"' WHERE 
CLAIMID='"+cid+"'";
         //log("Issuing SQL: " + updateSQL);
         st.executeUpdate(updateSQL); //step 3
         success = true;
      } catch (Exception e) {
         logError("Exception updating claim status: "+e);
         success = false;
      } finally { //step 4
         close(st);
         close(con);
      }
      return success;
   }
```

iSync

This section shows a snippet from the ITSO MobileAdjuster example to demonstrate how to develop a iSync Client application for DB2e. Before the application (syncer) can sync, you need to prepare the credentials (user ID and password), which are needed for initialization. In the Mobile Adjuster application, we capture the user ID and password credentials entered at the time of logon to the application and store them for retrieval to be used for the sync.

In Example 11-6 we show a sample of how to create the credential object.

Example 11-6 Sample code to prepare the credentials for db2e sync

```
   private Properties createProps() {
      Properties props = new Properties();
      props.put("isync.user", userid);
      props.put("isync.password", password);
      props.put("isync.trace", "detailed");
      props.put("isync.messagesize", "1000000");
      props.put("isync.timeout", ISync.TIMEOUT_MINIMUM);
      return props;
   }
```

After the credential object is created, we are ready to initialize the sync. The major steps needed during the initialization are as follows:

1. Get a DB2e sync provider.
2. Get an instance of the synchronization service from the provider.
3. Get an instance of the configuration store from the service object.
4. Get an instance of the synchronization driver from the configuration store object to perform synchronization.
5. Set the listener object for event notification from the syncer object during synchronization.

In Example 11-7 we demonstrate a snippet DatabaseSyncManager source code, which contains comments that identify the steps explained above.

Example 11-7 Sample code for db2e sync initialization

```
   private void initializeSync() {
      try {
         provider =
ISyncManager.getISyncProvider(ISyncManager.ISYNC_DB2E_PROTOCOL); //step 1
         service = provider.createSyncService(syncserver, createProps());
//step 2
         config = service.getConfigStore(localDatabaseLocation); //step 3
         syncer = config.getSyncDriver(); //step 4
         syncer.setSyncListener(this); //step 5
      } catch (ISyncException e) {
         logError("**** MobileAdjuster initialize sync exception ****");
         printISyncException(e);

logError("******************************************************\n");
      } catch (Exception e) {
         logError("**** MobileAdjuster initialize sync exception ****");
         logError(e.toString());

logError("******************************************************\n");
      }
   }
```

After initinialization, the syncer is ready to synchronize. In Example 11-8, the sync() method perform synchronization on all enabled subscription sets and checks the return code and exceptions for status of the synchronization.

Example 11-8 Sample code for DB2e sync

```
   public String sync() {
      log("host: " + syncserver + ", user: " + userid + ", pass: " + password
            + ", path: " + localDatabaseLocation);
      String resultMsg = "Data refresh result message";

      try {
         int rc = syncer.sync(); //step 1
         switch (rc) {
         case ISync.RTN_SUCCEEDED:
            log("Synchronization succeeded");
            resultMsg = "Data refreshed successfully";
            break;

         case ISync.RTN_CANCELED:
            //This occurs when system can resolve server name but can't
connect to it
            //Since there is no "Cancel" button in MobileAdjuster, this can
only be a timeout
            log("Synchronization canceled");
            resultMsg = "Data refresh timed out";
            break;

         default:
            log("Synchronization failed");
            resultMsg = "Data refresh failed";
            break;
         }
         ssArr = config.getSubscriptionSets(); //step 2
         for (int i = 0; i < ssArr.length; i++) {
            log("Subscription Set: " + ssArr[i].getName() + " Status: ");

            switch (ssArr[i].getStatus()) {
            case ISync.STATUS_READY:
               log("READY");
               break;

            case ISync.STATUS_COMPLETED:
               log("COMPLETED");
               break;

            case ISync.STATUS_CANCELED:
               log("CANCELED");
               break;

            default:
               log("FAILED");
               break;
            }
         }
      } catch (ISyncException e) {
         resultMsg = "Data refresh failed: "+translateISyncException(e);
         printISyncException(e);
```

```
        }
        return resultMsg;
    }
```

Summary to perform DB2e sync

To summarize a Java native synchronization application, the major steps are the following:

1. Import the DB2 Everyplace synchronization packages:

   ```
   import com.ibm.mobileservices.isync.*;
   import com.ibm.mobileservices.isync.event.*;
   ```

2. Implement the eventIssued method of the ISyncListener interface for event notification during synchronization.
3. Get an instance of the DB2 sync provider.
4. Get an instance of the synchronization service from the provider object.
5. Get an instance of the configuration store from the service object.
6. Get an instance of the synchronization driver from the configuration store object.
7. Register your application listener object that implements the ISyncListener interface for event notification from the synchronization driver object during synchronization.
8. Perform synchronization on all enabled subscription sets. Check the return code and exceptions for the status of the synchronization.
9. Close and free all resources of the synchronization.

11.2.5 Develop disconnectable app using Mobile Web Services

The focus of this section is to highlight how to develop a disconnectable Mobile Web Service Client application with the WebSphere Everyplace Client Toolkit.

The three classes shown in Table 11-5 will be generated by the Mobile Web Service Client wizard.

Table 11-5 Classes generated by the Mobile Web Services Client wizard

Class name	Package
ArrayOf_xsd_string	com.ibm.mobile.images
ManagerService	com.ibm.mobile.images
ManagerServiceSoap_Stub	com.ibm.mobile.images

Note:

- For details on how to deploy the ImageManager Web service used for the MobileAdjuster application, refer to 13.2.7, “Install the Image Manager Web service” on page 368.
- For more detailed information about Creating a Web Service application, please refer to one of the following redbooks:
  - *Develop Web Services applications* chapter in *Rational Application Developer V6 Programming Guide*, SG24-6449
  - *WebSphere Version 6 Web Services Handbook Development and Deployment*, SG24-6461

Deploy ImageManager enterprise application

The development of the Mobile Web Services Client application requires that the ImageManager enterprise application is deployed to a WebSphere Application Server V6.0 application server.

For details refer to 13.2.7, “Install the Image Manager Web service” on page 368.

Note: Ensure the Web service wsdl is available before creating the Mobile Web Service Client project by entering the following URL in a Web browser:

```
http://wedswin1.itso.ral.ibm.com/claimimages/wsdl/com/ibm/mobile/logic/image
s/ManagerService.wsdl
```

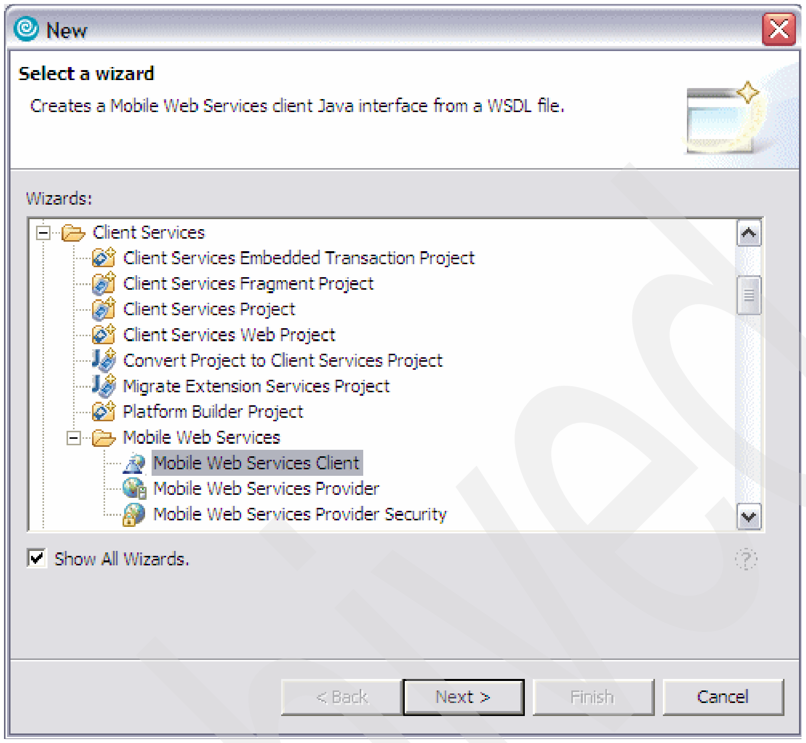
You should see wsdl XML in the Web browser if successful.

Create the Mobile Web Service Client

We can now run the Mobile Web Service Client project to generate the Web service stubs so that we can consume the Web service.

To create the Mobile Web Service Client project, do the following:

1. To launch a new Mobile Web Service Client Project wizard, select **File** → **New** → **Other**.
2. Expand **Client Services** → **Mobile Web Services**, and select **Mobile Web Services Client** (as seen in Figure 11-7 on page 302), and then click **Next**.


New

Select a wizard

Creates a Mobile Web Services client Java interface from a WSDL file.

Wizards:

- Client Services
  - Client Services Embedded Transaction Project
  - Client Services Fragment Project
  - Client Services Project
  - Client Services Web Project
  - Convert Project to Client Services Project
  - Migrate Extension Services Project
  - Platform Builder Project
  - Mobile Web Services
    - Mobile Web Services Client
    - Mobile Web Services Provider
    - Mobile Web Services Provider Security

[✓] Show All Wizards.

Figure 11-7 Create a Mobile Client Services Client

3. When the Mobile Web Services Client Stub Generator Wizard appears, enter the URL of WSDL location and the source folder where the client code will be stored. For example, we entered the following values, as seen in Figure 11-8 on page 303, to create Web Service Client codes.

   – Source Folder: /MobileAdjusterLogic/src

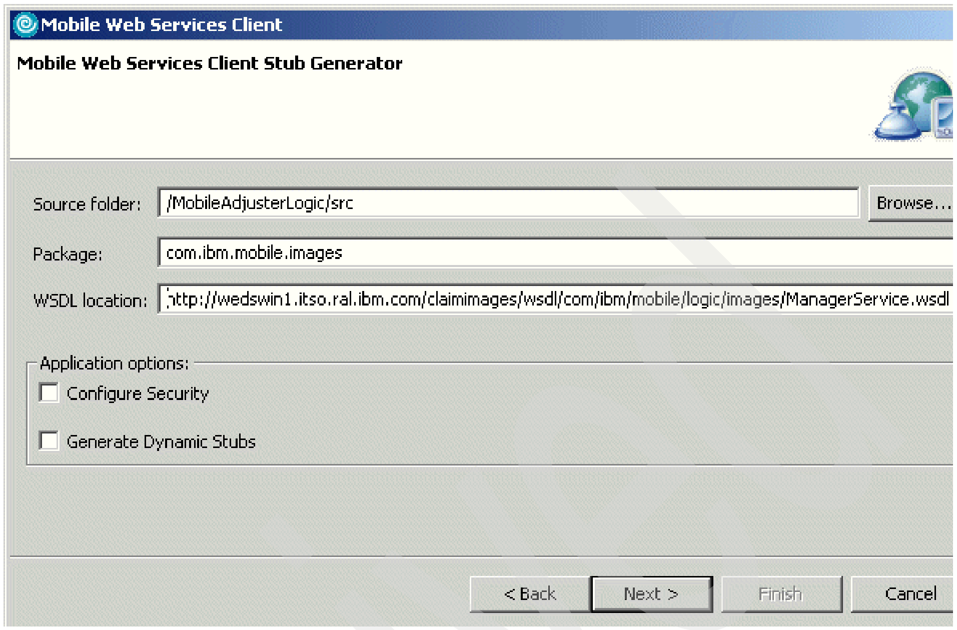
   – Package: com.ibm.mobile.images

     Note that entering the package is not required since the package is defined by the wsdl (server in the WSDL location must be available).

   – WSDL location:

     ```
     http://wedswin1.itso.ral.ibm.com/claimimages/wsdl/com/ibm/mobile/logic/i
     mages/ManagerService.wsdl
     ```

Note: The instance of the WebSphere Application Server with the Web Service provider application (that is, server1) must be running for the WSDL to be available. Otherwise an error will occur upon selecting **Next**.


Source folder: /MobileAdjusterLogic/src

Package: com.ibm.mobile.images

WSDL location: http://wedswin1.itso.ral.ibm.com/claimimages/wsdl/com/ibm/mobile/logic/images/ManagerService.wsdl

Application options:

- [ ] Configure Security
- [ ] Generate Dynamic Stubs

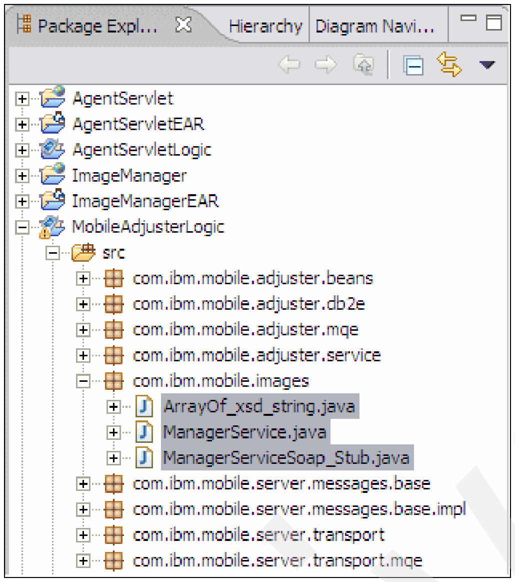
Mobile Web Services Client
Mobile Web Services Client Stub Generator
Browse...
< Back
Next >
Finish
Cancel

Figure 11-8   Mobile Web Services Client Stub Generator

4. When the Data Type Marshalling dialog appears, click **Finish**.

   When done, you should see the generated Web Service stubs under the com.ibm.mobile.images package, as seen in Figure 11-9 on page 304.

   > **Note:** The Data Type Marshalling dialog is used to create custom marshallers for types referenced in the WSDL that are incompatible with JSR 172. In this case there are none.


Package Expl... Hierarchy Diagram Navi...

```
AgentServlet
AgentServletEAR
AgentServletLogic
ImageManager
ImageManagerEAR
MobileAdjusterLogic
    src
        com.ibm.mobile.adjuster.beans
        com.ibm.mobile.adjuster.db2e
        com.ibm.mobile.adjuster.mqe
        com.ibm.mobile.adjuster.service
        com.ibm.mobile.images
            ArrayOf_xsd_string.java
            ManagerService.java
            ManagerServiceSoap_Stub.java
        com.ibm.mobile.server.messages.base
        com.ibm.mobile.server.messages.base.impl
        com.ibm.mobile.server.transport
        com.ibm.mobile.server.transport.mqe
```

Figure 11-9 Java files generated by Mobile Web Services Client wizard

Client Web Services application code

Example 11-9 demostrates how to instantiate the Web service stub and use it to invoke the Web service.

Example 11-9 Sample snippet AdjusterService.java to downloadImage from server

```
public boolean downloadImage(String fileName) {
      boolean success = false;
      String filePath = getImageFromRepository(fileName);
        log("Making web service call to download image...");
      ManagerService imgMgr = new ManagerServiceSoap_Stub();
      String webSvcHost = MobileConstants.get(MobileConstants.IMG_SERVERADDR);
      String webSvcPort = MobileConstants.get(MobileConstants.IMG_SERVERPORT);
      ((ManagerServiceSoap_Stub)imgMgr)._setProperty(
            javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,

"http://"+webSvcHost+":"+webSvcPort+"/claimimages/services/ManagerService");

      byte [] bytes = {};
      try {
         bytes = imgMgr.downloadImage(fileName);
         log("Successfully downloaded image with size "+bytes.length+"
bytes");
         FileOutputStream fos = new FileOutputStream(filePath);
         fos.write(bytes);
         success = true;
```

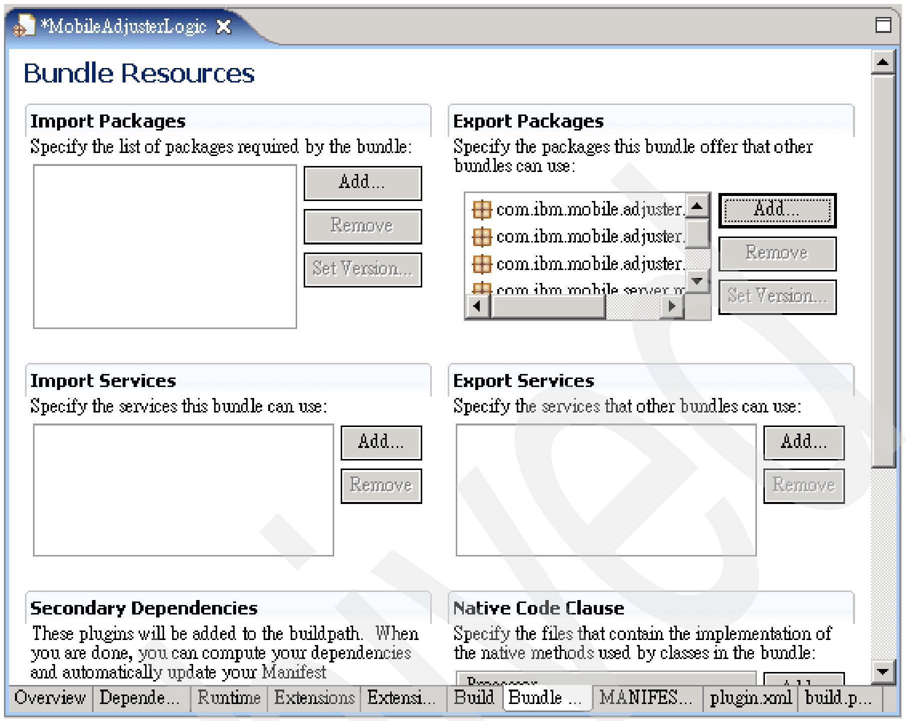
```
            fos.close();
         } catch (RemoteException e) {
            logError("Image Manager upload failed: "+e);
            e.printStackTrace();
            success = false;
         } catch (JAXRPCException e) {
            logError("Image Manager upload failed: "+e);
            e.printStackTrace();
            success = false;
         } catch (FileNotFoundException e) {
            logError("Failed to create new image file in file system:
"+filePath);
            logError(e.toString());
            success = false;
         } catch (IOException e) {
            logError("Failed to write data to new image file: "+filePath);
            logError(e.toString());
            success = false;
         }
         return success;
      }
```

11.2.6 Edit the MANIFEST.MF for exporting packages

Some application service bundles such as MQe need to call back into MobleAdjusterLogic bundle; therefore, we demonstrate how MobileAdjusterLogic exports packages containing some code that is needed by another bundle.

The steps for exporting packages are as follows:

1. Open the Java Perspective Package Explorer view.
2. Expand **MobileAdjusterLogic** → **META-INF**.
3. Double-click the **MANIFEST.MF** file to open it in the editor.
4. Click the **Bundle Resources** tab.
5. Click **Add** under Export Packages.
6. When the Select packages to export dialog appears, select the following bundles (Ctrl+Shift to select multiple), as seen in Figure 11-10 on page 306:
   - Add com.ibm.mobile.adjuster.beans.
   - Add com.ibm.mobile.adjuster.mqe.
   - Add com.ibm.mobile.adjuster.service.
   - Add com.ibm.mobile.server.messages.base.
   - Add com.ibm.mobile.server.transport.mqe.


*MobileAdjusterLogic

Bundle Resources

Import Packages
Specify the list of packages required by the bundle:

Export Packages
Specify the packages this bundle offer that other bundles can use:
com.ibm.mobile.adjuster.
com.ibm.mobile.adjuster.
com.ibm.mobile.adjuster.
com.ibm.mobile.server.m

Import Services
Specify the services this bundle can use:

Export Services
Specify the services that other bundles can use:

Secondary Dependencies
These plugins will be added to the buildpath. When you are done, you can compute your dependencies and automatically update your Manifest

Native Code Clause
Specify the files that contain the implementation of the native methods used by classes in the bundle:

Figure 11-10 Bundle Resources of MANIFEST.MF file for MobleAdjusterLogic

7. Click the **Runtime** tab.
8. Click **Add** under Library exporting.
9. Select the following bundles (Ctrl+Shift to select multiple):
   - Add com.ibm.mobile.adjuster.beans.
   - Add com.ibm.mobile.adjuster.mqe.
   - Add com.ibm.mobile.adjuster.service.
   - Add com.ibm.mobile.server.messages.base.
   - Add com.ibm.mobile.server.transport.mqe.
10. Save the modified MANIFEST.MF file.

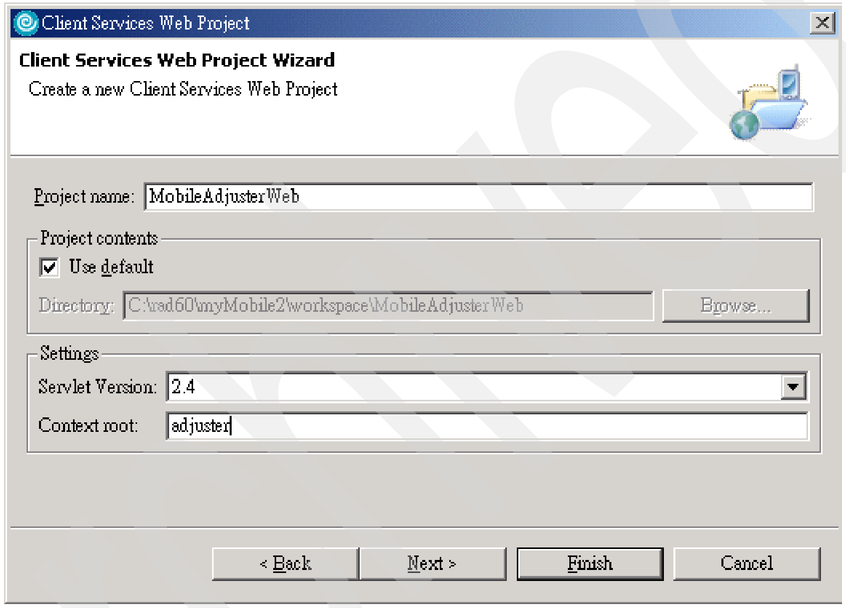
11.3 Develop disconnectable app with local Web UI

This section describes how to develop a disconnectable application with a local Web user interface.

11.3.1 Create the Client Services Web Project

To create the Client Services Web Project, do the following:

1. Select **File** → **New** → **Project**.
2. Expand **Client Services**, select **Client Services Web Project**, and then click **Next**.
3. When the Client Services Web Project dialog appears, enter `MobileAdjusterWeb` in the Project name field, enter `adjuster` in the Context root field (as seen in Figure 11-11), and then click **Next**.


Client Services Web Project Wizard
Create a new Client Services Web Project

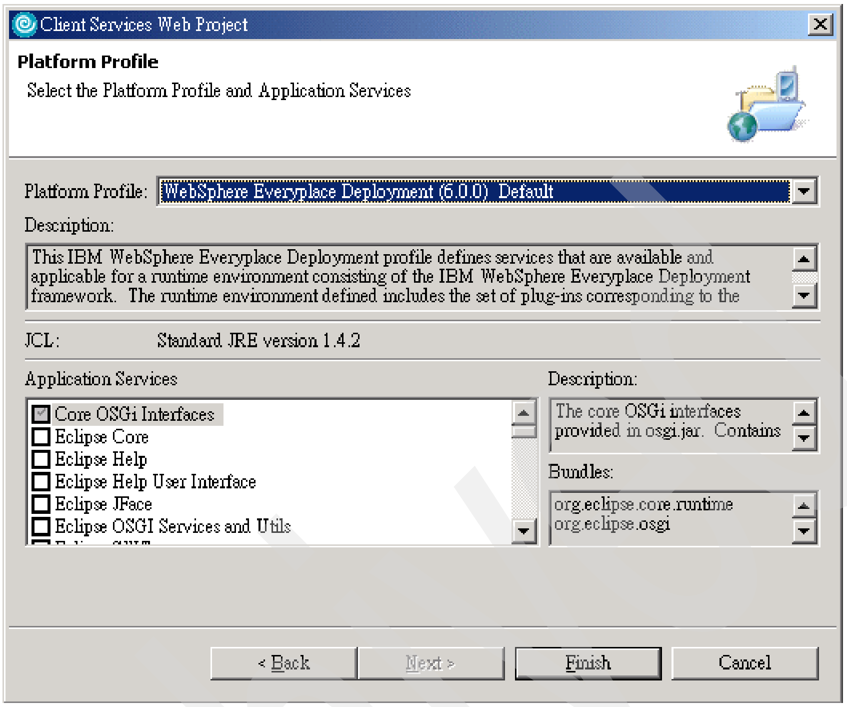
Project name: MobileAdjusterWeb

Project contents
☑ Use default
Directory: C:\rad60\myMobile2\workspace\MobileAdjusterWeb

Settings
Servlet Version: 2.4
Context root: adjuster

Figure 11-11 New Client Services Web Project wizard

4. When the Platform Profile dialog appears, do the following:
   a. Select **WebSphere Everyplace Deployment (WebSphere Everyplace Deployment(6.0.0) Default** from the Platform Profile drop-down list.
   b. When done the dialog should look like Figure 11-12 on page 308. Click **Finish**.


Client Services Web Project

Platform Profile

Select the Platform Profile and Application Services

Platform Profile: WebSphere Everyplace Deployment (6.0.0) Default

Description:

This IBM WebSphere Everyplace Deployment profile defines services that are available and applicable for a runtime environment consisting of the IBM WebSphere Everyplace Deployment framework. The runtime environment defined includes the set of plug-ins corresponding to the

JCL: Standard JRE version 1.4.2

Application Services

- [✓] Core OSGi Interfaces
- [ ] Eclipse Core
- [ ] Eclipse Help
- [ ] Eclipse Help User Interface
- [ ] Eclipse JFace
- [ ] Eclipse OSGI Services and Utils

Description:

The core OSGi interfaces provided in osgi.jar. Contains

Bundles:

org.eclipse.core.runtime
org.eclipse.osgi

Figure 11-12 Platform Profile dialog

The Web Project directory structure for the newly created MobileAdjusterWeb Project is displayed in Figure 11-13 on page 309.

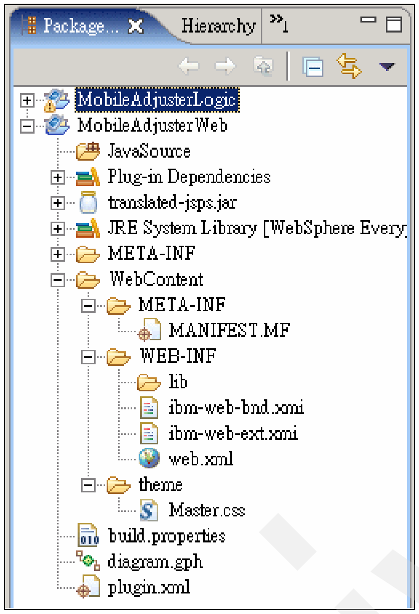


Figure 11-13 Client Services Web Project directory structure

– Java Source: This folder contains the project's java source code for regular classes, JavaBeans, and servlets. We create the servlets listed in Table 11-6.

Table 11-6 Servlets to add in MobileAdjusterWeb projects

Class name	Package
AdjusterServlet	com.ibm.mobile.adjuster.web
ConfigServlet	com.ibm.mobile.adjuster.web
ImageServlet	com.ibm.mobile.adjuster.web

– Web Content: This folder holds the contents of the WAR file that will be deployed to the server. It contains all the Web resources, including compiled Java classes and servlet, HTML files, JSPs, and graphics needed for the application.

11.3.2 Import servlet source code

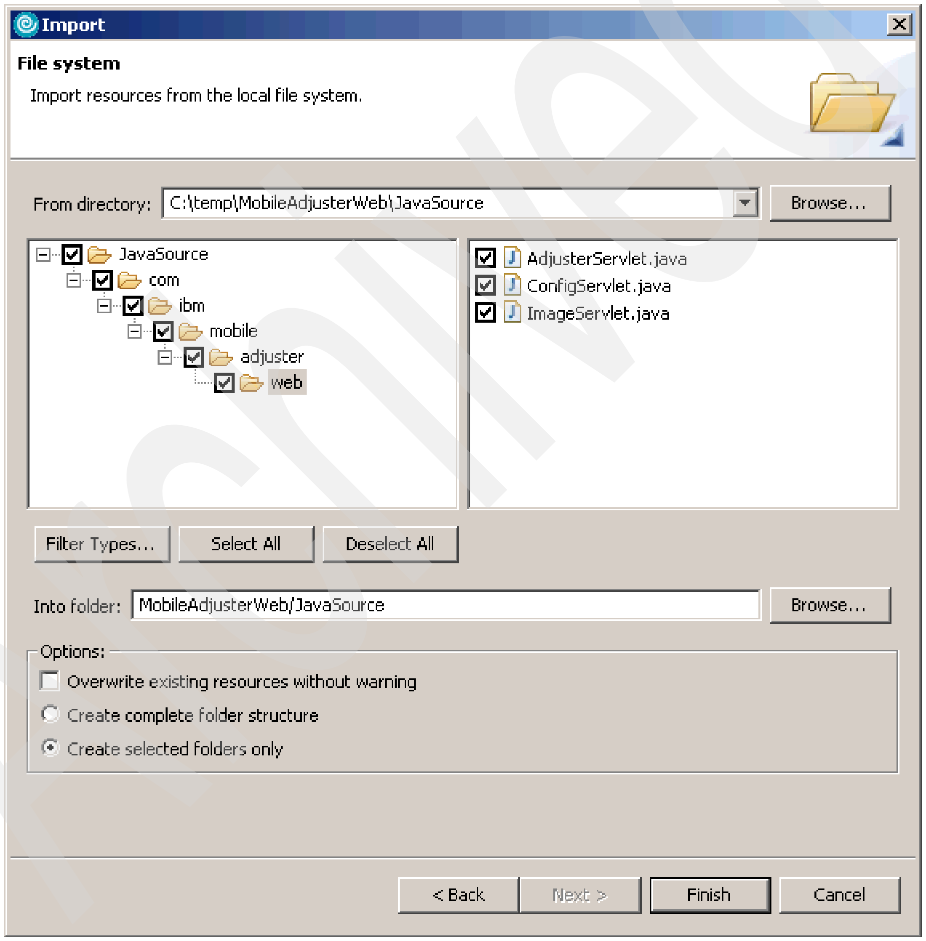
Import the servlet source code to expedite development:

1. Expand **MobileAdjusterWeb**, right-click the **JavaSource** folder, and select **Import**.

2. When the Import dialog appears, select **File system** and click **Next**.
3. When the File system dialog appears, do the following:
   a. Enter `c:\temp\MobileAdjusterWeb\JavaSource` in the From directory.

      This directory was created as a result of unzipping the ITSO_clientsrc.zip in 11.1.2, “Download ITSO sample code” on page 280.

   b. Check **JavaSource**.
   c. Click **Filter types**. Check **.java** files and click **OK**.
   d. When done, the dialog should look like Figure 11-14. Click **Finish**.



From directory: C:\temp\MobileAdjusterWeb\JavaSource

- ☑ JavaSource
  - ☑ com
    - ☑ ibm
      - ☑ mobile
        - ☑ adjuster
          - ☑ web

☑ AdjusterServlet.java
☑ ConfigServlet.java
☑ ImageServlet.java

Into folder: MobileAdjusterWeb/JavaSource

Options:
- ☐ Overwrite existing resources without warning
- ○ Create complete folder structure
- ● Create selected folders only

Figure 11-14 Import servlet source code

11.3.3 Import Web content source code

To import the Web content source code, do the following:

1. Expand **MobileAdjusterWeb**, right-click the **WebContent** folder, and select **Import**.
2. When the Import dialog appears, select **File system** and click **Next**.
3. When the File system dialog appears, do the following:
   a. Enter `c:\temp\MobileAdjusterWeb\WebContent` in the From directory.

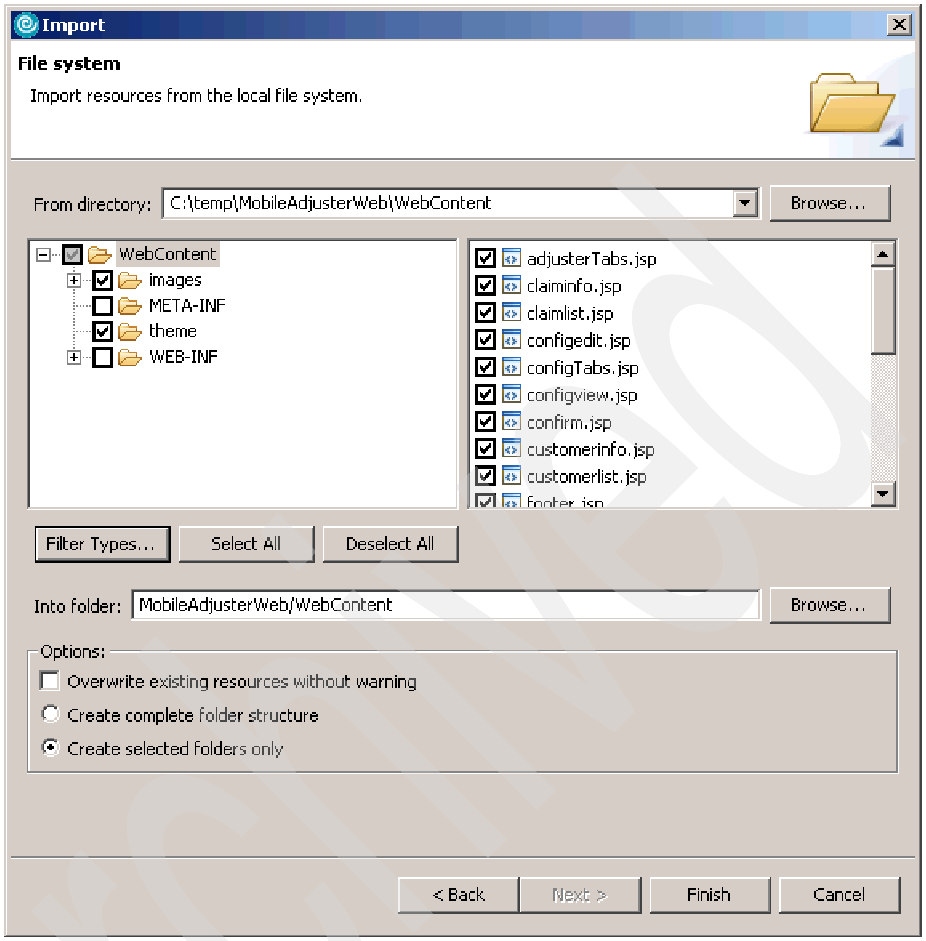
      This directory was created as a result of unzipping the ITSO_clientsrc.zip in 11.1.2, “Download ITSO sample code” on page 280.

   b. Check **JavaSource**.

   c. Uncheck **META-INF** and **WEB-INF**.

   d. Click **Filter types**. Check **.jsp** files and click **OK**.

   e. When done, the dialog should look like Figure 11-15 on page 312. Click **Finish**.


Import
File system
Import resources from the local file system.
Filter Types... Select All Deselect All
< Back Next > Finish Cancel

From directory: C:\temp\MobileAdjusterWeb\WebContent

- WebContent
  - [✓] images
  - [ ] META-INF
  - [✓] theme
  - [ ] WEB-INF

- [✓] adjusterTabs.jsp
- [✓] claiminfo.jsp
- [✓] claimlist.jsp
- [✓] configedit.jsp
- [✓] configTabs.jsp
- [✓] configview.jsp
- [✓] confirm.jsp
- [✓] customerinfo.jsp
- [✓] customerlist.jsp
- [✓] footer.jsp

Into folder: MobileAdjusterWeb/WebContent

Options:
- [ ] Overwrite existing resources without warning
- ( ) Create complete folder structure
- (●) Create selected folders only

Figure 11-15 Files imported for WebContent folder

4. When prompted to overwrite the Master.css, click **Yes**.

11.3.4 Add servlet to MobileAdjusterWeb deployment descriptor

To add the Servlet into MobileAdjusterWeb application deployment descriptor, do the following:

1. Expand **MobileAdjusterWeb** → **WebContent** → **WEB-INF**, and double-click **web.xml** to open in the editor.
2. Click the **Servlets** tab and click **Add** under Servlets and JSPs.
3. Add the servlets into the MobileAdjusterWeb application, as follows:
   a. Click **Add** next to URL Mappings.

b. Enter `/adjuster` for the URL mapping, as seen in Table 11-7, and then click **OK**.

c. Check **Use existing Servlet class** and click **Browse**.

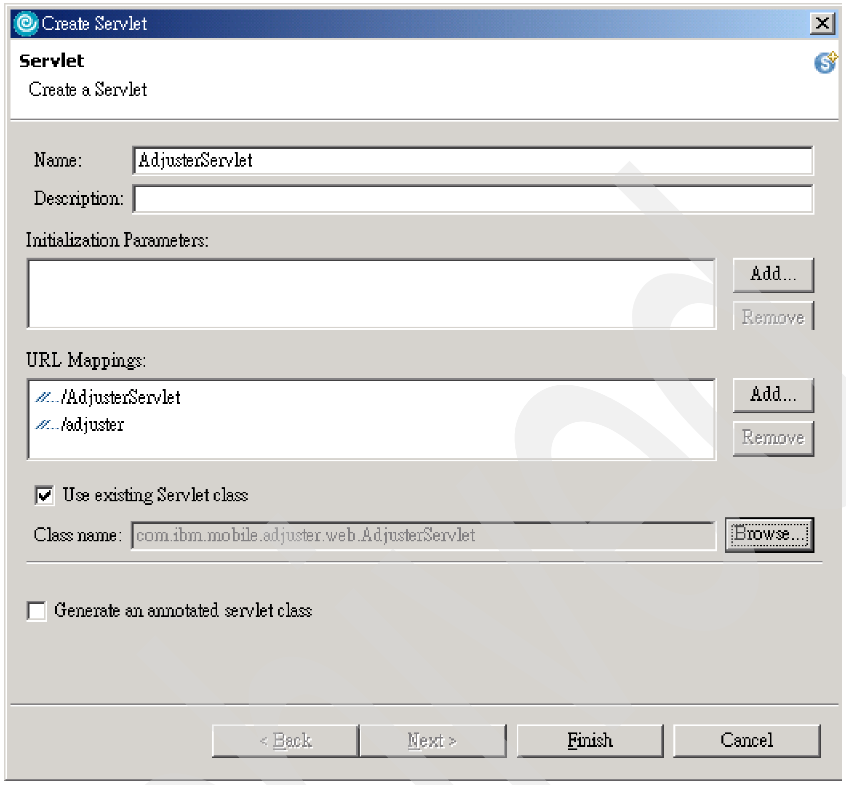
d. Enter the servlet name (for example, AdjusterServlet), select it from the list, and click **OK**.

e. When done, the dialog should look like Figure 11-16 on page 314.

Table 11-7 Add the servlets listed to MobileAdjusterWeb

Name	Class name	URL mapping
AdjusterServlet	com.ibm.mobile.web.AdjusterServlet	/AdjusterServlet /adjuster
ConfigServlet	com.ibm.mobile.web.ConfigServlet	/ConfigServlet /config
ImageServlet	com.ibm.mobile.web.ImageServlet	/ImageServlet /images

4. Repeat the above process for each servlet listed in Table 11-7.
5. Save the deployment descriptor.


Create Servlet

Servlet

Create a Servlet

Name: AdjusterServlet

Description:

Initialization Parameters:

URL Mappings:

/AdjusterServlet

/adjuster

[✓] Use existing Servlet class

Class name: com.ibm.mobile.adjuster.web.AdjusterServlet

[ ] Generate an annotated servlet class

Figure 11-16 Create a Servlet in Web deployment descriptor

11.3.5 Modify the MANIFEST.MF deployment descriptor for bundle

To modify the MANIFEST.MF deployment descriptor for the OSGi bundle, do the following:

1. Expand **MobileAdjusterWeb** → **META-INF** and double-click **MANIFEST.MF**.
2. Click the **Dependencies** tab.
   a. Click **Add**.
   b. Select **MobileAdjusterLogic** from the Required Plug-ins list and click **OK**. The resulting page should look like Figure 11-17 on page 315.

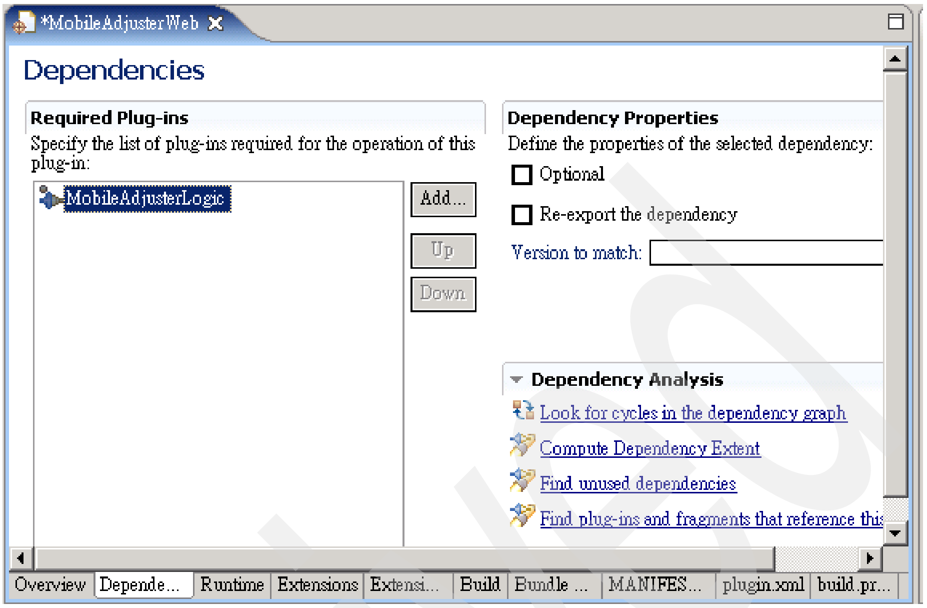


Figure 11-17 Add dependencies for MobileAdjusterWeb project

3. Click the **Extensions** tab.
   a. Click **Add** under All Extensions.
   b. Uncheck **Show only extension points from the required plug-ins**.
   c. Select **com.ibm.eswe.workbench.WctWebApplication**, as seen in Figure 11-18 on page 316, and click **Finish**.

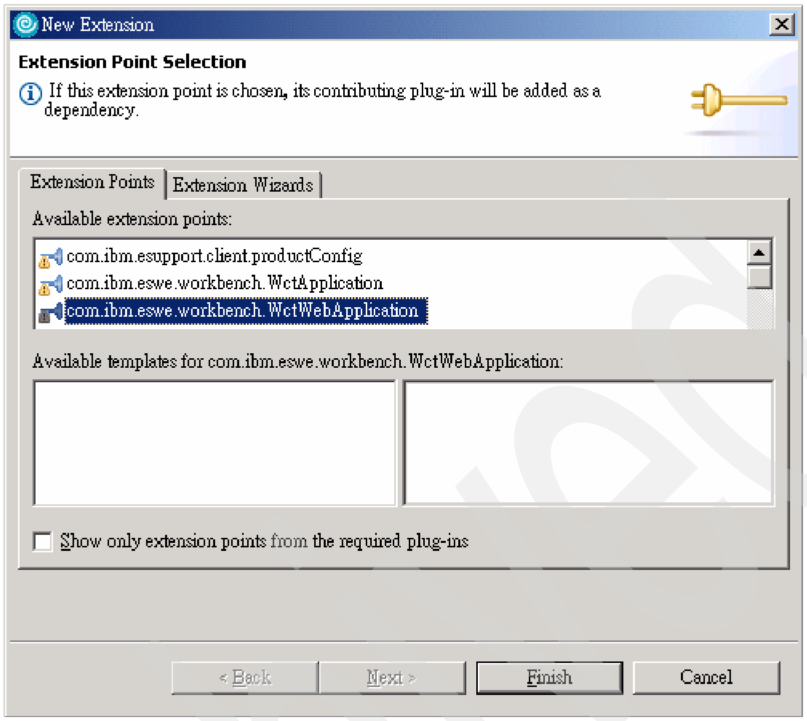
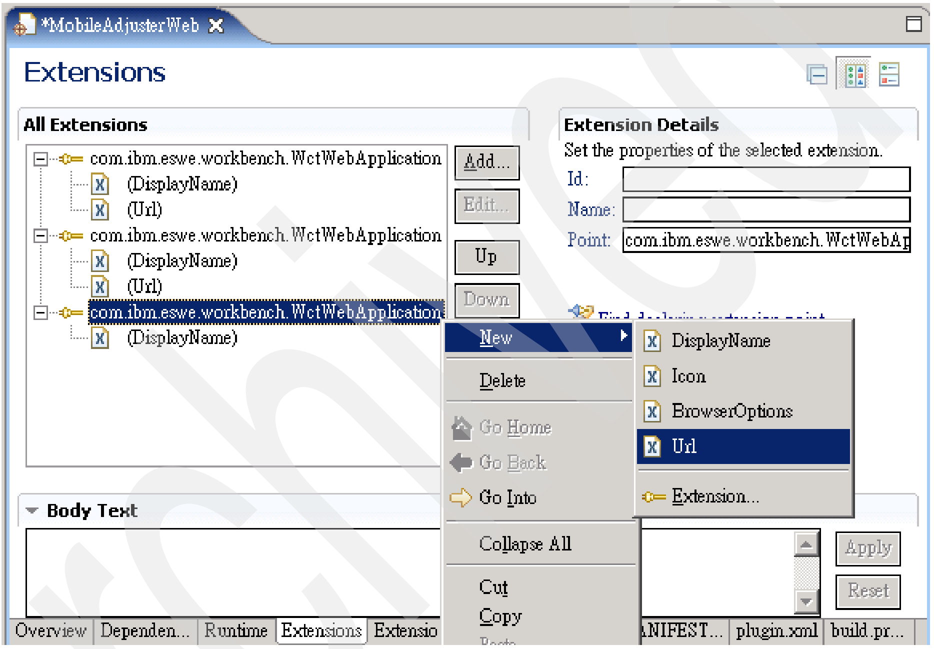

New Extension
Extension Point Selection
If this extension point is chosen, its contributing plug-in will be added as a dependency.
Extension Points
Extension Wizards
Available extension points:
com.ibm.esupport.client.productConfig
com.ibm.eswe.workbench.WctApplication
com.ibm.eswe.workbench.WctWebApplication
Available templates for com.ibm.eswe.workbench.WctWebApplication:
Show only extension points from the required plug-ins
< Back
Next >
Finish
Cancel

Figure 11-18 Add Extension points

4. Edit the extension file for the bundle.
   a. Right-click **com.ibm.eswe.workbench.WctWebApplication** and select **New** → **DisplayName**.
   b. Click **Body Text** to expand.
   c. Enter `Mobile Config` in the Text field and click **Apply**.
   d. Right-click **com.ibm.eswe.workbench.WctWebApplication** and select **New** → **URL** (see Figure 11-19 on page 317).
   e. Click **Body Text** to expand.
   f. Enter `/adjuster/config` in Text field and click **Apply**.
   g. In the local drop-down list, select **true**.
5. Repeat the above process to edit the bundles for the other entries in Table 11-8 on page 317.

Table 11-8 Extensions for bundle

Name	Display name	URL
ConfigServlet	Mobile Config	/adjuster/config
ImageServlet	Mobile Image	/adjuster/images
AdjusterServlet	Mobile Adjuster	/adjuster/adjuster



com.ibm.eswe.workbench.WctWebApplication
(DisplayName)
(Url)
com.ibm.eswe.workbench.WctWebApplication
(DisplayName)
(Url)
com.ibm.eswe.workbench.WctWebApplication
(DisplayName)

Id:
Name:
Point: com.ibm.eswe.workbench.WctWebAp

Figure 11-19 Add New URL

Figure 11-20 on page 318 displays the completed Extensions page.

6. Save the file.
7. Rebuild the project.

   Assuming Automatic Build is enabled, once you save the file it will build, and the remaining errors should disappear.

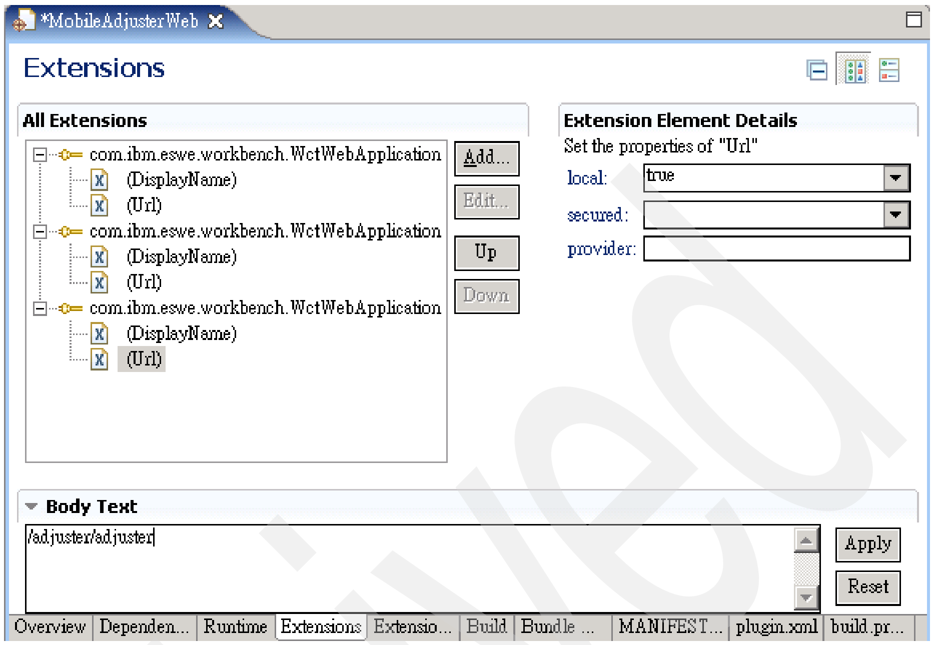


Figure 11-20 Completed extensions page

Example 11-10 shows the contents of the plugin.xml file with the actual extension point information.

Example 11-10 Sample plugin.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.0"?>
<plugin>
   <extension point="com.ibm.eswe.workbench.WctWebApplication">
         <DisplayName>Mobile Adjuster Config</DisplayName>
         <Url local="true">/adjuster/config</Url>
   </extension>
   <extension point="com.ibm.eswe.workbench.WctWebApplication">
         <DisplayName>Mobile Adjuster Images</DisplayName>
         <Url local="true">/adjuster/images</Url>
   </extension>
   <extension point="com.ibm.eswe.workbench.WctWebApplication">
      <DisplayName>Mobile Adjuster</DisplayName>
      <Url local="true">/adjuster/adjuster</Url>
   </extension>
</plugin>
```

11.3.6 Run the MobileAdjuster

For details on running the ITSO MobileAdjuster, refer to 10.6, “Create client configuration and run application” on page 274.

11.4 Package the application for deployment

This section describes how to package the server-side and client-side application artifacts in preparation for deployment to the WebSphere Everyplace Deployment Server and Client nodes.

11.4.1 Export server-side application for deployment

This section describes how to export the following server-side components of the ITSO MobileAdjuster application to deployable artifacts:

- Export the MobileAgent.ear.
- Export the MobileAdjusterMQeServer.jar.
- Export the ImageManager.ear.

Note: This section assumes that you have imported the ITSO_MobileAdjusterPI.zip profile interchange file containing the server projects. For details refer to 10.5, “Import the completed sample code” on page 270.

Export the MobileAgent.ear

To export the MobileAgent Web application to the MobileAgent.ear, do the following:

1. Select **File** → **Export** from the Rational Application Developer main menu.
2. When the Export dialog appears, select **EAR file** from the export destination type list, and then click **Next**.
3. When the EAR Export dialog appears, do the following:
   a. Select **AgentServletEAR** from the EAR project drop-down list.
   b. Enter the Destination location and file name; for example, we entered c:\temp\deploy\server\MobileAgent.ear.
   c. Check **Overwrite existing file**.
   d. Click **Finish** to create the EAR file.

Export the MobileAdjusterMQeServer.jar

To export the MobileAdjusterServer project to the MobileAdjusterMQeServer.jar, do the following:

1. Select **File** → **Export** from the main menu of Rational Application Developer.
2. When the Export dialog appears, select **JAR file** from the export destination type list, and then click **Next**.
3. When the JAR Export dialog appears, do the following:
   a. Check **MobileAdjusterServer** from the resources to export list.
   b. Enter the JAR file location and file name; for example, we entered c:\temp\deploy\server\MobileAdjusterMQeServer.jar.
   c. Check **Overwrite existing files without warning**.
   d. Click **Finish** to create the JAR file.

Export the ImageManager.ear

To export the ImageManagerEAR project to the ImageManager.ear, do the following:

1. Select **File** → **Export** from the main menu.
2. When the Export dialog appears, select **EAR file** from the export destination type list, and then click **Next**.
3. When the EAR Export dialog appears, do the following:
   a. Select **ImageManagerEAR** from the EAR project drop-down list.
   b. Enter the Destination location and file name; for example, we entered c:\temp\deploy\server\ImageManager.ear.
   c. Check **Overwrite existing file**.
   d. Click **Finish** to create the EAR file.

11.4.2 Create the client-side application Update Site zip file

This section describes how to create an Update Site zip file containing the client-side application components of the ITSO MobileAdjuster application for the purposes of application deployment.

Note: The Update Site zip can be used to install the application locally on the WED client or be further packaged using the OSGi NativeAppBundle tool to perform a remote software distribution job from DMS. For details refer to 14.2.6, “Software distribution” on page 396.

In our example, the creation of an Update Site is a two-step process. We first create a feature, and then create an Update Site containing this feature:

1. Ensure that the MobileAdjusterWeb and MobileAdjusterLogic projects have no compilation errors within the Rational Application Developer workspace.
2. Create a new feature:
   a. Select **File** → **New** → **Other** from the main menu.
   b. Expand **Plug-in Development**, select **Feature Project**, and then click **Next**.
   c. When the Feature Name dialog appears, enter `MobileAdjusterFeature` in the Project Name field and then click **Next**.
   d. When the Feature Properties dialog appears, we entered the following and then clicked **Next**:
      - Feature ID: MobileAdjusterFeature (default)
      - Feature Name: MobileAdjuster Feature (default)
      - Feature Version: 6.0.0 (We entered.)

      Note: The Feature Version is used primarily for displaying the version (for example, Device Manager Console or WED Client). Programatically, it used when you search for applications from an Eclipse remote Update Site.

   e. When the Referenced Plug-ins and Fragments dialog appears, check **MobileAdjusterWeb** and **MobileAdjusterLogic**, and then click **Finish**.
   f. You may be prompted to open the Plug-in Development perspective (if not already opened). Click **Yes**.
3. Create a new Update Site.
   a. Select **File** → **New** → **Other** from the main menu.
   b. Expand **Plug-in Development**, select **Update Site Project**, and then click **Next**.
   c. When the Update Site Project dialog appears, enter `MobileAdjusterUpdateSite` in the Project name field and then click **Finish**.

      After clicking Finish, the site.xml for the Update Site will be opened.
   d. In the site.xml editor, click **Add** in the Feature to Build column.
   e. Check **MobileAdjusterFeature**, and then click **Finish**.
   f. Click **New Category** under Features to Publish.
   g. When the New Category Definition dialog appears, enter `Mobile Adjuster Client Application` in the Name and Label fields, and then click **OK**.

h. Add the MobileAdjusterClient feature to the to the MobileAdjuster Client Application category:

   i. Right-click **MobileAdjusterFeature** and select **Publish**. You should now see the feature listed under Features to Publish.

   ii. Drag the **MobileAdjusterFeature** under the Features to Publish column, on top of the **Mobile Adjuster Client Application** category. This adds the feature to the category.

i. Save the site.xml file by selecting **File** → **Save** (or by pressing Ctrl+S).

j. Click **Build All** in site.xml editor under the Features to Build column to generate the Update Site contents on the file system.

k. Verify from the file system that application jar files exist. Navigate to the c:\workspace\MobileAdjusterUpdateSite folder.

   You should see something like the files listed in Example 11-11.

Example 11-11 MobileAdjuster Update Site files and directory structure

```
c:\workspace\MobileAdjusterUpdateSite\site.xml
c:\workspace\MobileAdjusterUpdateSite\features\MobileAdjusterFeature_6.0.0.jar
c:\workspace\MobileAdjusterUpdateSite\plugins\MobileAdjusterLogic_1.0.0.jar
c:\workspace\MobileAdjusterUpdateSite\plugins\MobileAdjusterWeb_1.0.0.jar
```

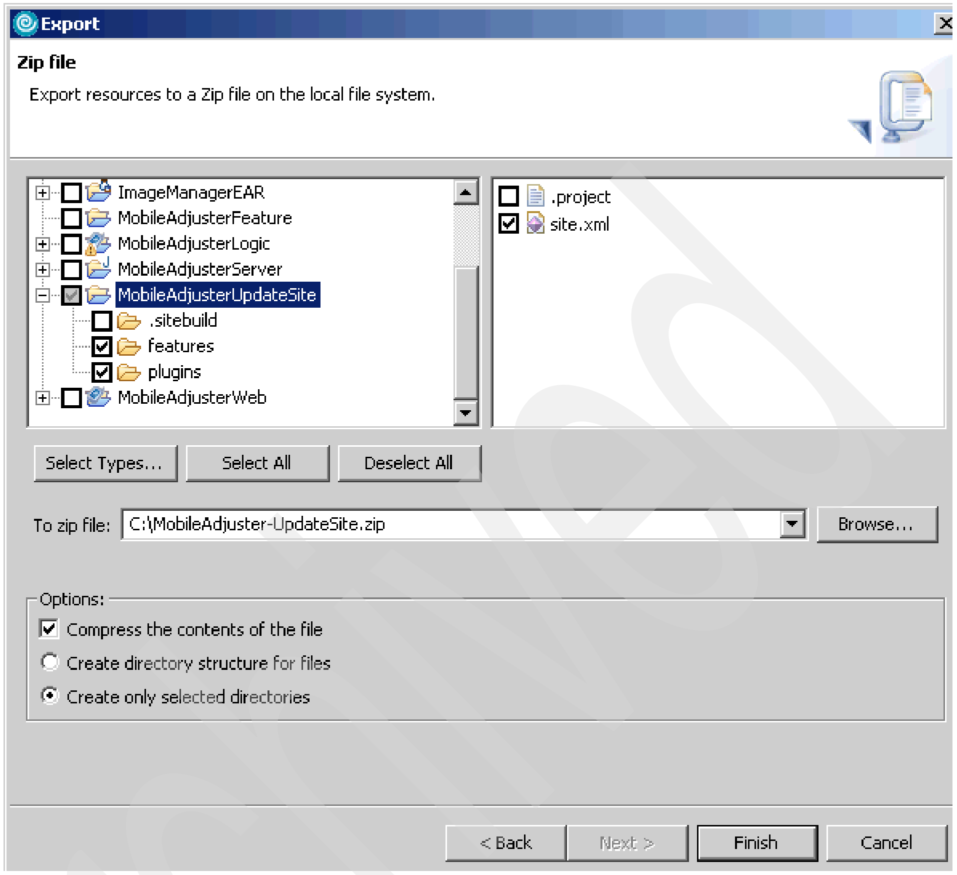
4. Create the MobileAdjuster-UpdateSite.zip file.

   a. Right-click the **MobileAdjusterUpdateSite** project in the Package Explorer view and select **Export**.

   b. Select **Zip File** and click **Next**.

   c. When the Zip File dialog appears, do the following:

      i. Verify that MobileAdjusterUpdateSite is checked.
      ii. Uncheck .project.
      iii. Expand MobileAdjusterUpdate Site.
      iv. Uncheck the .sitebuild folder.
      v. Enter `C:\MobileAdjuster-UpdateSite.zip` in the To zip file field.
      vi. Select the **Create only selected directories** radio button.
      vii. Click **Finish**.



Export resources to a Zip file on the local file system.

- [ ] ImageManagerEAR
- [ ] MobileAdjusterFeature
- [ ] MobileAdjusterLogic
- [ ] MobileAdjusterServer
- [✓] MobileAdjusterUpdateSite
  - [ ] .sitebuild
  - [✓] features
  - [✓] plugins
- [ ] MobileAdjusterWeb

- [ ] .project
- [✓] site.xml

To zip file: C:\MobileAdjuster-UpdateSite.zip

Options:
- [✓] Compress the contents of the file
- ○ Create directory structure for files
- ● Create only selected directories

Figure 11-21 Zip file dialog to export the MobileAdjuster-UpdateSite.zip

12

Runtime environment installation

This chapter describes how to implement the runtime environment by node for the ITSO working example. The procedures are high level and intended to be used in conjunction with the product installation guides.

The value add of the procedures found in this chapter are three fold. First, we provide high-level installation and configuration procedures of the software components by node for our scenario. Second, the procedure includes sample values that put in context the information found in the product installation guides. Third, the procedures include best practices, tips, and workarounds based on our first-hand experiences.

The chapter is organized into the following sections:

- Planning and scenario overview
- Access Server node installation
- Access Client node installation
- Directory Server node installation (optional)
- Web Server Redirector node installation (optional)

12.1 Planning and scenario overview

This section describes the runtime topology used for the ITSO working example. In addition, we provide information regarding the hardware and software levels we used to implement the runtime environment.

The ITSO example runtime topology is based largely on the Distributed Rich Client::Runtime pattern::Product mapping (see 7.5, "Distributed Rich Client::Runtime pattern" on page 169).

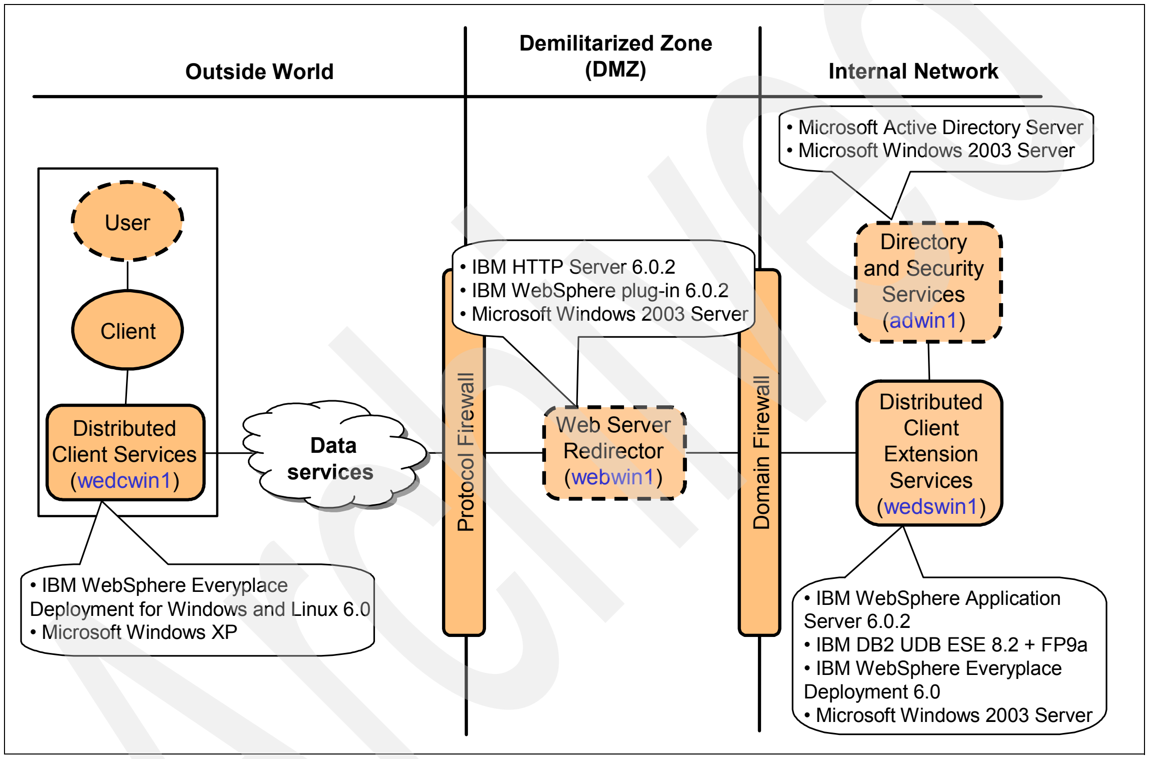

Outside World
Demilitarized Zone (DMZ)
Internal Network
User
Client
Distributed Client Services (wedcwin1)
Data services
Protocol Firewall
• IBM HTTP Server 6.0.2
• IBM WebSphere plug-in 6.0.2
• Microsoft Windows 2003 Server
Web Server Redirector (webwin1)
Domain Firewall
• Microsoft Active Directory Server
• Microsoft Windows 2003 Server
Directory and Security Services (adwin1)
Distributed Client Extension Services (wedswin1)
• IBM WebSphere Everyplace Deployment for Windows and Linux 6.0
• Microsoft Windows XP
• IBM WebSphere Application Server 6.0.2
• IBM DB2 UDB ESE 8.2 + FP9a
• IBM WebSphere Everyplace Deployment 6.0
• Microsoft Windows 2003 Server

Figure 12-1 ITSO working example Product mapping

12.1.1 Hardware used within the ITSO runtime environment

We used the following hardware to implement the nodes of the ITSO runtime environment on the Microsoft Windows 2003 Server platform.

Note: For simplicity in referencing the nodes, we use the following names:

- Access Server node (Distributed Client Extension Services)
- Access Client node (Distributed Client Services)
- Directory Server node (Directory and Security Services)
- Web Server Redirector node (Web Server Redirector)

- Access Server node
  - IBM eServer™ xSeries® BladeCenter® HS20
    - 1 CPU, Intel Xeon® 2.4 GHz
    - 2 GB RAM
    - 40 GB Hard Disk
    - 1 Ethernet Adapter
    - Host name: wedswin1.itso.ral.ibm.com
- Access Client node
  - IBM Thinkpad T40
    - 1 CPU, Intel Pentium M 1.5 GHz
    - 1 GB RAM
    - 40 GB Hard Disk
    - 1 Ethernet Adapter
    - Host name: wedcwin1.itso.ral.ibm.com
- Directory Server node
  - IBM eServer xSeries 230 (8658-61Y)
    - 1 CPU, Intel PIII 1 GHz
    - 512 MB RAM
    - 18 GB Ultra™ SCSI Hard Disk
    - 1 Ethernet Adapter
    - Host name: adwin1.itso.ral.ibm.com
- Web Server Redirector node
  - IBM eServer xSeries 230 (8658-61Y)
    - 1 CPU, Intel PIII 1 GHz
    - 512 MB RAM
    - 18 GB Ultra SCSI Hard Disk
    - 1 Ethernet Adapter
    - Host name: webwin1.itso.ral.ibm.com

Tip: VMWare Workstation V5.0

While developing the ITSO runtime environment, we used VMWare Workstation V5.0 extensively.

12.1.2 Software used within the ITSO runtime environment

The ITSO Access Client working example was implemented using the software levels listed in each of the following tables by node.

Table 12-1 Access Server node

Software	Version
Microsoft Windows 2003 Server	2003 + SP1 + critical fixes
IBM WebSphere Application Server	6.0.2 **Note**: 6.0 + 6.0.2 Refresh
IBM DB2 UDB, Enterprise Server Edition	8.1.9.917 **Note**: 8.2 + Fixpack 9a
IBM WebSphere Everyplace Deployment	6.0

Table 12-2 Access Client node

Software	Version
Microsoft Windows XP	XP + SP2 + critical fixes
IBM WebSphere Everyplace Deployment Client for Windows and Linux	6.0

Table 12-3 Directory Server node (optional)

Software	Version
Microsoft Windows 2003 Server, Standard Edition	2003 + SP1 + critical fixes
Microsoft Active Directory	2003 + SP1 + critical fixes

Table 12-4 Web Server Redirector node (optional)

Software	Version
Microsoft Windows 2003 Server, Standard Edition	2003 + SP1 + critical fixes
IBM HTTP Server	6.0.2
WebSphere Application Server - IBM HTTP Server plug-in **Note:** Once the IBM HTTP Server and plug-in are configured on this node, the IBM HTTP Server on the Access Server node can be disabled.	6.0.2

12.2 Access Server node installation

This section outlines the steps we followed to install the software components on the Access Server node.

Note: For more detailed information refer to the following product installation guides:

- IBM DB2 Universal Database V8.2
- IBM WebSphere Application Server V6.0
- IBM WebSphere Everyplace Deployment V6.0

12.2.1 Windows 2003 Server and system preparation

At the time of writing, IBM WebSphere Everyplace Deployment V6.0 only supported Microsoft Windows 2003 Server (Standard and Enterprise Edition).

Complete the following tasks:

- Windows 2003 Server installation.
- Verify network configuration.
- Ensure Microsoft Internet Information Server (IIS) is disabled.
- Ensure firewall is disabled during the installation.
- Create Windows administrative user ID.
- Assign Windows user right assignments.
- Log on as admin user ID during installation.

Windows 2003 Server installation

For our example, we installed Microsoft Windows 2003 Server, Standard Edition with Service Pack 1 and critical fixes.

Verify network configuration

Verify that the network configuration for your system is working properly before proceeding. This includes verifying that the host name and fully qualified host name can be resolved using ping and nslookup command-line utilities.

Ensure Microsoft Internet Information Server (IIS) is disabled

Ensure that the Microsoft Internet Information is disabled from Windows services to avoid TCP/IP port conflicts with the IBM HTTP Server.

Ensure firewall is disabled during the installation

Prior to installation, ensure that you do not have a firewall enabled to avoid conflicts with TCP/IP ports used by WebSphere Everyplace Deployment or supporting products.

Create Windows administrative user ID

During the installation of WebSphere Everyplace Deployment, you need to be logged in to the system with a Windows administrative user ID.

We created an administrative user ID named *admin*.

Assign Windows user right assignments

Ensure the following user right assignments are assigned from within the Windows Local Security Settings for the administrative user ID (for example, admin) used during installation:

- Act as part of the operating system.
- Adjust memory quotas for a process.
- Create a token object.
- Log on as a service.
- Replace a process level token.

You can access the Windows Local Security Settings by selecting **Start** → **Control Panel** → **Administrative Tools** → **Local Security Policy**. The User Right Assignments can be accessed by expanding **Security Settings** → **Local Policies** → **User Right Assignment**.

Log on as admin user ID during installation

Prior to installing WebSphere Everyplace Deployment and prerequisite software, log on to the system with the administrator ID you have assigned the proper user right assignments (see “Assign Windows user right assignments” on page 330).

12.2.2 DB2 Universal Database V8.2.2 installation

As a prerequisite to IBM WebSphere Everyplace Deployment V6.0, this section outlines the steps to install IBM DB2 Universal Database V8.2 and Fix Pack 9a.

Complete the following tasks:

- Install DB2 UDB V8.2.
- Install DB2 UDB V8.2 Fixpack 9a.
- Verify DB2 UDB V8.2.2.

Install DB2 UDB V8.2

We have listed the key options we selected during the IBM DB2 Universal Database V8.2, Enterprise Edition installation:

1. Run setup.exe to start the installer from the root of the DB2 Universal Database CD.
2. Click **Install Product** to begin the installation.
3. We selected **Typical** for the installation type.
4. We accepted the default options for the remaining installation dialogs unless noted.
5. We installed DB2 Universal Database in the C:\IBM\SQLLIB directory.
6. During the installation you will be prompted to create a DB2 Administration Server user. We accepted the default (`db2admin`).
7. When the installation is complete the DB2 Product Updates dialog and First Steps dialogs appear.
   a. Click **No** on the DB2 Product Updates dialog.
   b. Click **Exit** on the First Steps dialog.

Install DB2 UDB V8.2 Fixpack 9a

We have listed the key options we selected during the IBM DB2 Universal Database V8.2 Fix pack 9a installation:

1. Download IBM DB2 Universal Database V8.2 Fix pack 9a from:

   http://www.ibm.com/software/data/db2/udb/support/downloadv8_windows32bit.html
2. Run **FP9a_WR21350_ESE.exe**, which is a self-extracting installer.
3. Click **Install Product** to begin the fix pack installation.
4. We accepted the default options for the remaining installation dialogs unless noted.
5. The Fix Pack installer will prompt you to stop the necessary DB2 services. Click **Yes**.
6. You will be prompted to restart your system for the changes to take effect. Click **Yes**.

   Note: You may be prompted to check for DB2 Universal Database Product Updates. Click **No**.
7. After the system restarts, the DB2 Product Updates dialog and First Steps dialogs appears.

a. Click **No** on the DB2 Product Updates dialog.
b. Click **Exit** on the First Steps dialog.

Verify DB2 UDB V8.2.2

After installing IBM DB2 Universal Database V8.2 and Fix Pack 9a, we recommend that you verify the DB2 Universal Database environment:

1. Open a DB2 command window and enter the following command:

   ```
   db2level
   ```

   It should return the internal level 8.1.9.917 for DB2 Universal Database V8.2 with Fix pack 9a.

2. Launch the DB2 Universal Database First Steps by clicking **Start** → **Programs** → **IBM DB2** → **Set-up Tools** → **First Steps**.
3. Create a sample database by clicking **Create Sample Database** in the First Steps dialog.
4. Click **Work with Databases** in the First Steps dialog. This launches the Control Center. Verify that the SAMPLE database exists.

12.2.3 WebSphere Application Server V6.0.2 installation

As a prerequisite to IBM WebSphere Everyplace Deployment V6.0, this section outlines the steps to install IBM WebSphere Application Server V6.0 with Refresh 2 (v6.0.2).

Complete the following tasks:

1. Install WebSphere Application Server V6.0.
2. Install IBM HTTP Server V6.0.
3. Install WebSphere V6.0 plug-in.
4. Verify WebSphere Application Server V6.0.
5. Install WebSphere Application Server V6.0.2 Refresh.
6. Install IBM HTTP Server V6.0.2 Refresh.
7. Install WebSphere V6.0.2 Plugin Refresh.
8. Verify WebSphere Application Server V6.0.2.

Install WebSphere Application Server V6.0

We have listed the key options we selected during the WebSphere Application Server V6.0 installation:

1. Run **launchpad.bat** to start the installer from the root of the WebSphere Application Server CD.
2. Click **Launch the installation wizard for WebSphere Application Server installation** from the launchpad base home page.

3. We installed the WebSphere Application Server to the C:\IBM\WebSphere\AppServer directory.
4. We selected **Full installation**.

> **Note:** There are a couple of reasons you might choose Custom installation instead of Full installation:
>
> - Custom installation allows you to deselect sample applications and Javadocs, which you may not want installed.
> - Custom installation allows you to define the ports used by WebSphere Application Server.

5. We accepted the default options for the remaining installation dialogs.

Install IBM HTTP Server V6.0

We have listed the key options we selected during the IBM HTTP Server V6.0 installation:

1. Run **launchpad.bat** to start the installer from the root of the WebSphere Application Server CD.
2. Click **Launch the installation wizard for IBM HTTP Server installation** from the launchpad base home page.
3. We accepted the default options for the remaining installation dialogs unless noted.
4. We installed the IBM HTTP Server to the C:\IBM\HTTPServer directory.
5. We selected **Typical** for the installation type.
6. We used the admin ID created in “Create Windows administrative user ID” on page 330, as the user ID to run IBM HTTP Server as a Windows service.

Install WebSphere V6.0 plug-in

We have listed the key options we selected during the Web Server (WebSphere V6.0) plug-in installation:

1. Run **launchpad.bat** to start the installer from the root of the WebSphere Application Server CD.
2. Click **Launch the installation wizard for Web Server plug-in installation** from the launchpad base home page.
3. We accepted the default options for the remaining installation dialogs unless noted.
4. We selected the **IBM HTTP Server V6** plug-in.

5. We selected **WebSphere Application Server machine (local)** since the IBM HTTP Server and WebSphere Application Server are on the same node.
6. We installed the WebSphere Application Server plug-ins to the C:\IBM\WebSphere\Plugins directory.
7. When prompted for the httpd.conf, provide the path by either entering in the text field or using the Browse button (for example, C:\IBM\HTTPServer\conf\httpd.conf).

Verify WebSphere Application Server V6.0

After the WebSphere Application Server, the IBM HTTP Server, and the plug-in have been installed, we recommend that you verify the installation before continuing.

1. Launch the WebSphere Application Server First Steps by clicking **Start** → **Programs** → **IBM WebSphere** → **Application Server v6** → **Profiles** → **default** → **First Steps**.
2. Verify the installation.
   a. Click **Installation verification** in the First Steps dialog.

      If successful, you should see the following message:

      ```
      IVT Verification Succeeded.
      Installation Verification is complete.
      ```

   b. Close the message dialog.
3. Verify the WebSphere Application Server Administrative Console:
   a. Click **Administrative Console** or enter the following URL to access the WebSphere Application Server Administrative Console:

      ```
      http://<was_hostname>:9060/ibm/console
      ```

   b. Enter a user ID and click **Log in**.

      At this point WebSphere Security is not enabled; thus, any ID can be used (for example, admin).

   c. After navigating the Administrative Console, click **Logout**.

- Verify snoop application is running properly:
   a. Ensure the following servers are started:
      - WebSphere Application Server - server1

        ```
        cd \IBM\WebSphere\AppServer\profiles\default\bin
        startserver server1
        ```

      - IBM HTTP Server Windows service

   b. Enter the following URL to verify the snoop application directly on WebSphere Application Server (no plug-in):

```
http://<was_hostname>:9080/snoop
```

   c. Enter the following URL to verify the snoop application using the plug-in:

```
http://<was_hostname>/snoop
```

- Stop the servers:

   a. Stop WebSphere Application Server.

```
cd \IBM\WebSphere\AppServer\profiles\default\bin
stopserver server1
```

   b. Stop the IBM HTTP Server Windows service.

Install WebSphere Application Server V6.0.2 Refresh

We have listed the key options we selected during the IBM WebSphere Application Server V6.0.2 Refresh installation.

Attention: If WebSphere Application Server Pack 2 is unpacked into the install_root directory, where there is an existing updateinstaller directory present, installation of Refresh Pack 2 indicates a successful installation when it actually fails to upgrade the Application Server.

To resolve this issue, you must delete the existing update installer directories for base WebSphere Application Server, IBM HTTP Server, and WebSphere Plugins.

For more information, refer to the following tech note:

```
http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&q1=java.lang.No
SuchMethodError&q2=FileActionPlugin&uid=swg21211850&loc=en_US&cs=utf-8&lang=
en
```

Note: If you have just installed IBM WebSphere Application Server V6.0 and no refresh levels (no updateinstaller directory) exist, you can simply unpack the Refresh (updateinstaller directory) to the root of WebSphere Application Server.

1. Ensure that the WebSphere Application Server is not running by doing one of the following:

   – Stop the WebSphere Application Server V6 - <hostname>Node01 Windows service.

   – Stop server1 from the command line:

```
cd \IBM\WebSphere\AppServer\profiles\default\bin
serverstatus - all
```

If server1 is started, enter the following to stop the server:

```
stopserver server1
```

2. Download the WebSphere Application Server V6.0.2 Refresh to a temporary directory. The Refresh (Intel Application Server) can be found at the following URL:

   http://www.ibm.com/support/docview.wss?rs=180&uid=swg24009813

3. Unzip the 6.0-WS-WAS-WinX32-RP0000002.zip to the root of the WebSphere Application Server installation directory. For example, after the unzip you will have a directory structure like the following:

   ```
   C:\IBM\WebSphere\AppServer\updateinstaller
   ```

4. Navigate to the updateinstaller directory (for example, C:\IBM\WebSphere\AppServer\updateinstaller) and run update.exe to start the WebSphere Update Installer.
5. We accepted the default options for the remaining installation dialogs unless noted.
6. When prompted, enter the Directory Name for WebSphere Application Server; for example, we entered C:\IBM\WebSphere\AppServer.
7. Select **Install maintenance package** and then click **Next**.
8. Enter the file name of the maintenance pack and click **Next** (for example, C:\IBM\WebSphere\AppServer\updateinstaller\maintenance\6.0-WS-WAS-WinX32-RP0000002.pak).
9. The maintenance package should be listed. Click **Next** to begin the update.

Note: The WebSphere Application Server V6.0.2 Refresh includes updates to the JDK™. The WebSphere Update Installer is a Java-based application that uses the JDK. The Update Installer will make a copy of the JDK files and restart using the copied version, thus allowing the JDK used by WebSphere Application Server to be updated (files not in use).

Ensure that you relaunch the Update Installer to install the V6.0.2 Refresh.

Install IBM HTTP Server V6.0.2 Refresh

We have listed the key options we selected during the IBM HTTP Server V6.0.2 Refresh installation:

1. Ensure the following IBM HTTP Server Windows services are stopped:
   - IBM HTTP Administration 6.0
   - IBM HTTP Server 6.0

2. Download the IBM HTTP Server V6.0.2 Refresh to a temporary directory. The Refresh (Intel IBM HTTP Server) can be found at:

   `http://www.ibm.com/support/docview.wss?rs=180&uid=swg24009813`

3. Unzip the 6.0-WS-WASIHS-WinX32-RP0000002.zip to the root of the IBM HTTP Server directory. For example, after the unzip you will have a directory structure like the following:

   ```
   C:\IBM\HTTPServer\updateinstaller
   ```

4. Navigate to the updateinstaller directory (for example, C:\IBM\IBMHTTPServer\updateinstaller) and run update.exe to start the WebSphere Update Installer.
5. We accepted the default options for the remaining installation dialogs unless noted.
6. When prompted, enter the Directory Name for IBM HTTP Server; for example, we entered C:\IBM\HTTPServer.
7. Select **Install maintenance package** and then click **Next**.
8. Enter the file name of the maintenance pack and click **Next** (for example, C:\IBM\HTTPServer\updateinstaller\maintenance\6.0-WS-WASIHS-WinX32-RP0000002.pak).
9. The maintenance package should be listed. Click **Next** to begin the update.

Install WebSphere V6.0.2 Plugin Refresh

We have listed the key options we selected during the WebSphere Plug-in V6.0.2 Refresh installation:

1. Ensure the following IBM HTTP Server Windows services are stopped:
   – IBM HTTP Administration 6.0
   – IBM HTTP Server 6.0
2. Download the WebSphere Plug-in V6.0.2 Refresh to a temporary directory. The Refresh (Intel Plug-ins) can be found at:

   `http://www.ibm.com/support/docview.wss?rs=180&uid=swg24009813`

3. Unzip the 6.0-WS-WASPlugIn-WinX32-RP0000002.zip to the root of the WebSphere plug-in's directory. For example, after the unzip you will have a directory structure like the following:

   ```
   C:\IBM\WebSphere\Plugins\updateinstaller
   ```

4. Navigate to the updateinstaller directory (for example, C:\IBM\WebSphere\Plugins\updateinstaller) and run update.exe to start the WebSphere Update Installer.
5. When prompted, enter the Directory Name for WebSphere Plug-ins; for example, we entered C:\IBM\WebSphere\Plugins.

6. Select **Install maintenance package** and then click **Next**.
7. Enter the file name of the maintenance pack and click **Next** (for example, C:\IBM\WebSphere\Plugins\updateinstaller\maintenance\6.0-WS-WASPlugIn-WinX32-RP0000002.pak).
8. The maintenance package should be listed. Click **Next** to begin the update.
9. We accepted the default options for the remaining installation dialogs.

Verify WebSphere Application Server V6.0.2

Now that WebSphere Application Server, IBM HTTP Server, and WebSphere Plugin are at V6.0.2 level, we recommend that you verify WebSphere Application Server before proceeding.

For details refer to "Verify WebSphere Application Server V6.0.2" on page 338.

12.2.4 WebSphere Everyplace Deployment V6.0 installation

This section describes the high-level steps to install IBM WebSphere Everyplace Deployment V6.0.

Complete the following tasks:

1. Install WebSphere Everyplace Deployment V6.0.
2. Verify WebSphere Everyplace Deployment installation.
3. Start WebSphere Everyplace Deployment servers.
4. Verify Everyplace Deployment User Management Console.

Install WebSphere Everyplace Deployment V6.0

We have listed the key options we selected during theIBM WebSphere Everyplace Deployment V6.0 installation:

1. Before installing WebSphere Everyplace Deployment, ensure you are logged in as the administrative user ID prepared in "Windows 2003 Server and system preparation" on page 329 (for example, admin).
2. Run launchpad.bat to start the installer from the root of the WebSphere Everyplace Deployment CD.
3. Click **Install the product** from the launchpad base home page.
4. Select the desired language for the installation (for example, English) and click **OK**.
5. When the Welcome dialog appears, click **Next**.
6. When the License Agreement dialog appears, review the terms and, if in agreement, select **I accept the terms in the license agreement** and then click **Next**.

7. The the Prerequisite software installed location dialog appears. Verify that the installer has properly detected the installation directories for DB2 Universal Database, IBM HTTP Server, and WebSphere Application Server, and then click **Next**.
8. When the WebSphere Application Server related information dialog appears, we accepted the default options and clicked **Next**.
9. We accepted the default installation options unless noted.
10. When prompted for the WebSphere Everyplace Deployment installation directory, we entered C:\IBM\WebSphere and then clicked **Next**.

> **Note:** The DB2Everyplace, DMS18, EveryplaceDeployment, and MQe directories will be created off the ..\WebSphere directory.

11. When the Specify Host Information for WebSphere Everyplace Deployment dialog appears, we entered the following and then clicked **Next**:
    - Host name: wedswin1.itso.ral.ibm.com (Ensure you enter the fully qualified host name.)
    - Create WebSphere Everyplace Deployment Administrator: wedadmin
    - Create WebSphere Everyplace Deployment Password: <password>
12. When the Database Login Information for WebSphere Everyplace Deployment dialog appears, we entered the following and then clicked **Next**:
    - Database Administrator ID: db2admin
    - Database Administrator Password: <password>
13. When the Installation Summary dialog appears, review the selections and then click **Next**.

> **Note:** This installation takes approximately 1–2 hours, depending on the hardware specs of your system.

14. When the installation is complete, click **Finish**.

Verify WebSphere Everyplace Deployment installation

Verify WebSphere Everyplace Deployment installation:

1. To launch the WebSphere Everyplace Deployment First Steps, click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **First Steps**.
2. Click **Verify Installation** from the First Steps page.

You should see the following message if successful:

```
Verification of WebSphere Everyplace Deployment Completed Successfully.
```

Note: WebSphere Everyplace Deployment displays the results of these tasks and logs the results into the wed_ivt.log, which resides in the WebSphere Everyplace Deployment installation\logs directory (for example, C:\IBM\WebSphere\EveryplaceDeployment\logs).

If you get errors regarding any of the verification steps, refer to the Troubleshooting the installation section in the WebSphere Everyplace Deployment InfoCenter.

Start WebSphere Everyplace Deployment servers

The WebSphere Everyplace Deployment servers are application servers that run within WebSphere Application Server. They can be started as a group or individually.

To start the WebSphere Everyplace Deployment servers, do the following:

1. Ensure the IBM HTTP Server Windows service is started.
2. To start all WebSphere Everyplace Deployment servers, click **Start** → **All Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **Start Servers**.

   Note: After WebSphere Everyplace Deployment is installed, WebSphere security is enabled; thus, a valid administrator user ID and password are required to stop the application servers.

   To stop all WebSphere Everyplace Deployment servers, click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **Stop Servers**.

   You will be prompted to enter the WebSphere Everyplace Deployment administrator ID and password.

   Or to start the WebSphere Everyplace Deployment servers individually, do the following:

   a. Open a command window and navigate to the following directory:

   ```
   C:\IBM\WebSphere\AppServer\profiles\default\bin
   ```

   b. Enter the following command to check the status of the servers:

   ```
   serverstatus -all
   ```

You should see a status list for each of the following application servers:

```
coreServices
DMS_AppServer
IBMDB2eServer
```

c. Enter the following command to start the application servers:

```
startserver coreServices
```

d. Repeat the process for each of the application servers listed.

3. To start server1 do one of the following:

   – Click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Application Server** → **Start Servers**.

   – To start server1 from the command line, enter the following:

```
cd C:\IBM\WebSphere\AppServer\profiles\default\bin
startserver server1
```

Note: To stop server1, enter the following:

```
stopserver server1 -username wedadmin -password password
```

Verify Everyplace Deployment User Management Console

The WebSphere Everyplace Deployment User Management Console is used to manage users and groups (create, edit, delete). The Console interacts with the WebSphere Member Manager sub system.

To verify the WebSphere Everyplace Deployment User Management Console, do the following:

1. To start the WebSphere Everyplace Deployment User Management Console, click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **User Management Console**.

   Alternatively, enter the following URL in a Web browser:

```
http://wedswin1.itso.ral.ibm.com/user-management
```

2. Click **Log out** when done.

12.3 Access Client node installation

This section outlines the steps we followed to install the software components on the Access Client node. The primary software component on this node is the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux, which in this example will be installed on the Windows XP platform.

12.3.1 Windows XP installation

For our example, we installed Microsoft Windows XP Professional, Service Pack 2, and critical fixes.

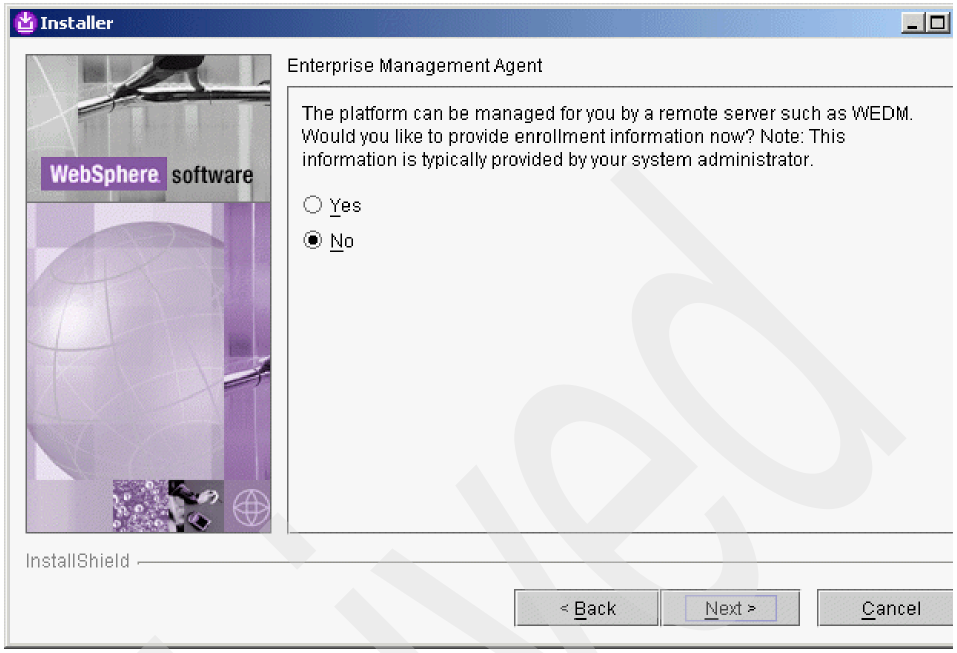
12.3.2 WebSphere Everyplace Deployment for Windows installation

To install IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (aka Client), do the following:

1. Insert the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux CD.
2. Navigate to the install directory and run setupwin32.exe to start the installer.
3. When the Welcome dialog appears, click **Next**.
4. When the License Agreement dialog appears, review the terms and, if in agreement, select **I accept the terms in the license agreement** and then click **Next**.
5. When the Installation directory dialog appears, enter the desired install directory and then click **Next** (for our example, we entered C:\IBM\WebSphere). The components will be installed into folders off the root.
6. When the Setup Type dialog appears, you can select from either Minimal or Full and then click **Next** (for our example, we selected **Full**).
7. When the Enterprise Management Agent dialog appears, as seen in Figure 12-2 on page 343, we accepted the default (No) and clicked **Next**.

Note: There a few key issues we would like to address about the dialog displayed in Figure 12-2 on page 343:

- WEDM in this case is WebSphere Everyplace Device Manager.
- The WebSphere Everyplace Deployment Server also includes the ability to manage remote platforms (WebSphere Everyplace Deployment for Windows and Linux, WebSphere Client Technology Micro Edition, WebSphere Client Technology Micro Edition Enterprise Offering). By selecting Yes, you can configure the client to be managed by a WebSphere Everyplace Deployment Server.
- The agent component for remote management is installed regardless of the selection (Yes/No). By selecting Yes, you can configure the client platform to the server in a subsequent dialog.
- The configuration can be performed post installation; thus, we selected **No** on the dialog.



Enterprise Management Agent

The platform can be managed for you by a remote server such as WEDM. Would you like to provide enrollment information now? Note: This information is typically provided by your system administrator.

Yes [✓ No]

Figure 12-2 Enterprise Management Agent

8. When the Installation summary dialog appears, review the selections and then click **Next** to begin copying files.
9. When the Provisioning platform components dialog appears, click **Next**.
10. When prompted to launch WebSphere Everyplace Deployment when the install is complete, select **No** and then click **Next**.
11. When the Summary dialog appears, you should see a message that the installation was successful. Click **Finish**.

12.4 Directory Server node installation (optional)

The Directory Server node is optional. By default, WebSphere Everyplace Deployment Member Manager used for user management is configured to use DB2 Universal Database. It can optionally be configured to use Microsoft Active Directory Server (LDAP).

Note: IBM WebSphere Everyplace Deployment V6.0 only supports Microsoft Active Directory. The Active Directory must be installed on a separate node from WebSphere Everyplace Deployment.

For implementation details, refer to the chapter titled “Migration to LDAP” in *IBM WebSphere Everyplace Deployment V6.0, Installation and Administration*, SG24-7141.

12.5 Web Server Redirector node installation (optional)

For an added level of security, the Web Server Redirector node can be placed in the DMZ. This configuration is optional for the ITSO working example runtime environment.

For implementation details, refer to the chapter titled “Add a Remote Web Server” in *IBM WebSphere Everyplace Deployment V6.0, Installation and Administration*, SG24-7141.

13 Application deployment

Applications that follow the Distributed Rich Client application pattern, such as the ITSO example, will have application assets to be deployed on the server and the client platform. A key advantage of server-managed clients is that the client-side application can be installed and managed remotely from a Device Manager server software distribution job.

This chapter provides a procedure for deploying the application assets of the ITSO client access working example. In some cases we reference other sections of this book for details, and in others we have included the details in this chapter. It is important that you follow the sequence of the steps in this chapter due to some application and configuration dependencies.

This chapter is organized into the following sections:

- Prerequisites for application deployment
- Server-side application deployment
- Client-side application deployment

13.1 Prerequisites for application deployment

This section describes the prerequisites for deploying the ITSO Mobile Adjuster sample application.

13.1.1 Download the ITSO sample code zip

Appendix E, “Additional material” on page 503, describes how to download and unpack the ITSO sample code for this book.

After unpacking the ITSO sample code 6775code.zip, you will have a C:\6775code directory containing the ITSO MobileAdjuster sample source files and deployable artifacts.

13.1.2 Runtime environment

The application deployment requires two nodes with the following installed:

- Access Server node: IBM WebSphere Everyplace Deployment V6.0
- Access Client node: IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux

For details on installing the runtime environment used for this book, refer to Chapter 12, “Runtime environment installation” on page 325.

13.1.3 Server startup

The WebSphere Everyplace Deployment servers are application servers that run within WebSphere Application Server. They can be started as a group or individually.

The WebSphere Everyplace Deployment servers are application servers that run within WebSphere Application Server. They can be started as a group or individually.

To start the WebSphere Everyplace Deployment servers, do the following:

1. Ensure the IBM HTTP Server Windows service is started.
2. To start all WebSphere Everyplace Deployment servers, click **Start** → **All Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **Start Servers**.

Note: After WebSphere Everyplace Deployment is installed, WebSphere security is enabled; thus, a valid administrator user ID and password will be required to stop the application servers.

To stop all WebSphere Everyplace Deployment servers, click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **Stop Servers**.

You will be prompted to enter the WebSphere Everyplace Deployment administrator ID and password.

Or, to start the WebSphere Everyplace Deployment servers individually, do the following:

a. Open a command window and navigate to the following directory:

```
C:\IBM\WebSphere\AppServer\profiles\default\bin
```

b. Enter the following command to check the status of the servers:

```
serverstatus -all
```

You should see a status list for each of the following application servers:

```
coreServices
DMS_AppServer
IBMDB2eServer
```

c. Enter the following command to start the application servers:

```
startserver coreServices
```

d. Repeat this process for each of the application servers listed.

3. To start server1 do one of the following:

– Click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Application Server** → **Stop Servers**.

– To start server1 from the command line, enter the following:

```
cd C:\IBM\WebSphere\AppServer\profiles\default\bin
startserver server1
```

Note: To stop server1, enter the following:

```
stopserver server1 -username wedadmin -password password
```

13.2 Server-side application deployment

This section describes the application deployment steps required to deploy the ITSO server-side application assets to the runtime environment.

Complete the following tasks:

1. Create and populate the application database.
2. Create users and groups.
3. Create the DB2e sync subscriptions.
4. Install the MQe Claim Server application.
5. Install the Web application services.
6. Install the MobileAgent enterprise application.
7. Install the Image Manager Web service.

13.2.1 Create and populate the application database

To create and populate the application database, do the following on the Access Server node:

1. Open a command window and navigate to the C:\6775code\deploy\server\database directory.
2. Enter the following command to create the application databases and populate the databases with sample data:

   ```
   mobadj_create_db.bat . false db2admin password
   ```

Script syntax:

```
mobadj_create_db.bat <dirpath> <dropflag> <dbuserid> <dbpassword>
```

- <dirpath> is the directory path of where the scripts are located. You can just specify a period (.) if you are in the directory where the scripts are located.
- <dropflag> is a value of true or false. If true, the script will attempt to clean up the existing MobileAdjuster database.
- <dbuserid> is an existing operating system user ID that will be the owner of the database objects and will be used for connecting to the database. Note that this user ID must exist in order for this script to function correctly.
- <dbpassword> is the password that is associated with <dbuserid> and will be used for connecting to the database. Note that this password must be the correct password associated with the user ID identified by <dbuserid> in order for this script to function correctly.

This script will output all actions performed to a logfile in the directory local directory where the script is run into a file called mobadj_create_db.log.

3. After the database script has run, check the mobadj_create_db1.log and mobadj_create_db2.log files for errors.
4. Open a DB2 command window by clicking **Start** → **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**.
5. Verify that the databases have been created by entering the following commands in a DB2 command window:

   ```
   db2 list db directory
   ```

   You should see the following application databases listed:

   ```
   MOBADJ
   M_MOBADJ
   ```

6. Verify the databases where populated with data by entering the following commands in a DB2 command window:

   ```
   db2 connect to MOBADJ user db2admin using password
   db2 select * from CUSTOMERS
   ```

   You should see a list of sample customers displayed.

Note: As an alternative to using the DB2 command-line interface, you can use the graphical DB2 Control Center by clicking **Start** → **Programs** → **IBM DB2** → **General Administration** → **Control Center**.

13.2.2 Create users and groups

This section describes how to create the users and groups listed in Table 13-1 for the ITSO sample application using the WebSphere Everyplace Deployment User Management Console.

Table 13-1 Application users and groups

User ID	First name	Last name	Group
adjuster1	Joe	Adjuster	DB2eAdjusters
devadjuster1 **Note:** This user is only needed for development testing.	Joe	Adjuster	DB2eAdjusters

Restriction: At the time of writing, a unique user ID was required for each device used by DB2 Everyplace. In our example, we have a WED Client node in the runtime and a WED Client in the development environment both sharing the same WED Server. The user ID that we use log in to the MobileAdjuster application must be unique; thus, we created the devadjuster1 user ID to be used while testing in the development environment.

Launch the User Management Console

To launch the WebSphere Everyplace Deployment User Management Console, do the following:

1. Ensure the WebSphere Everyplace Deployment Servers are started. Refer to 13.1.3, “Server startup” on page 346 for details.
2. Enter the following URL in a Web browser to access the User Management Console:

   ```
   http://wedswin1.itso.ral.ibm.com/user-management/login.jsp
   ```

   Note: Alternatively, if on the system where WebSphere Everyplace Deployment is installed, click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **User Management Console**.

3. When the Log In page appears, enter the Administrator ID (for example, wedadmin) and password (as seen in Figure 13-1 on page 351), and then click **Log In**.

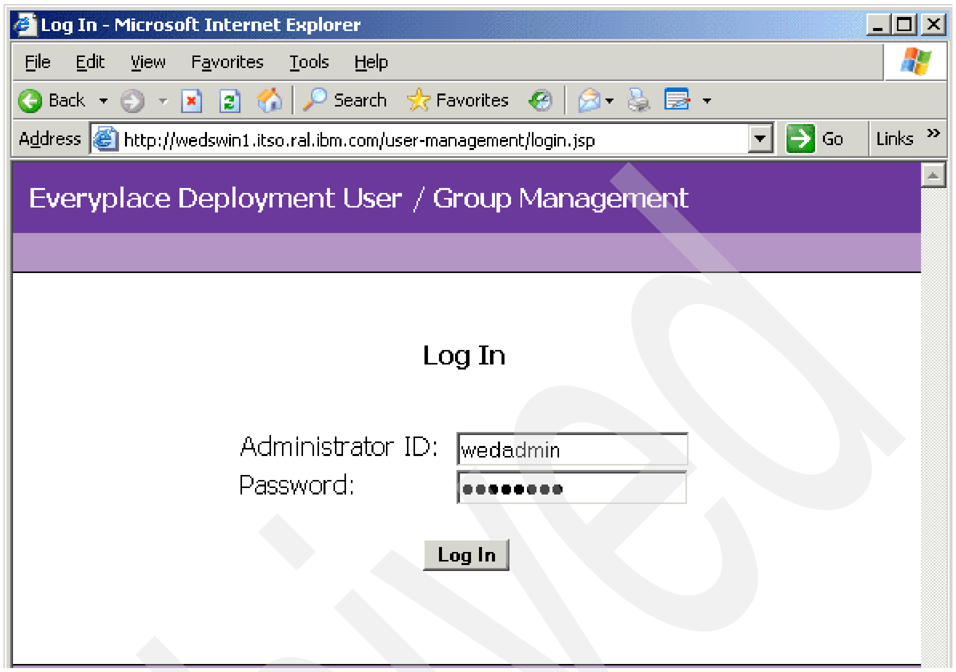
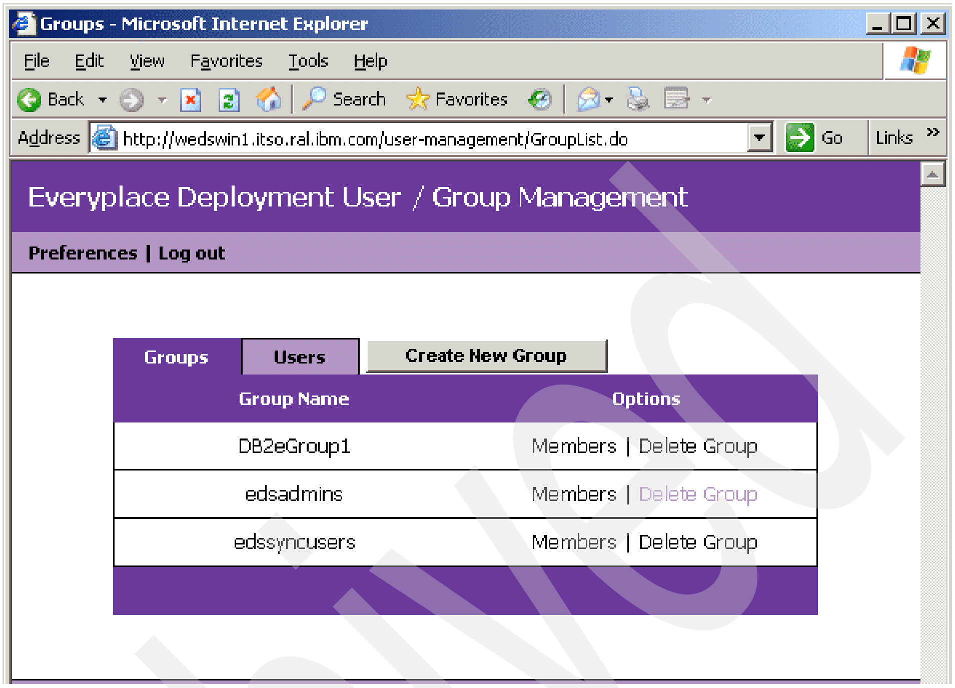

Log In - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://wedswin1.itso.ral.ibm.com/user-management/login.jsp Go Links
Everyplace Deployment User / Group Management
Log In
Administrator ID: wedadmin
Password: ●●●●●●●●●
Log In

Figure 13-1 User Management Console - Log In page

Create groups

To create groups in the WebSphere Everyplace Deployment User Management Console, do the following:

1. Click the **Groups** tab from the User Management Console.
2. Click the **Create New Group** button, as seen in Figure 13-2 on page 352.


Groups - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://wedswin1.itso.ral.ibm.com/user-management/GroupList.do
Go Links

Everyplace Deployment User / Group Management

Preferences | Log out

Groups | Users | Create New Group

Group Name	Options
DB2eGroup1	Members | Delete Group
edsadmins	Members | Delete Group
edssyncusers	Members | Delete Group

Figure 13-2 User Management Console home page

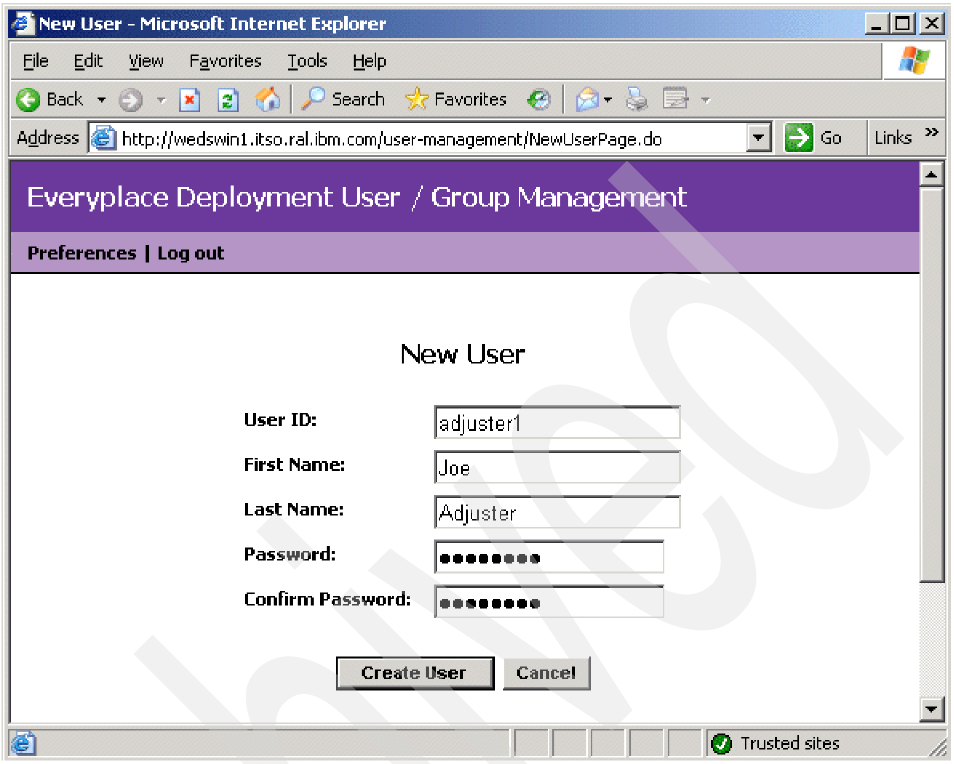
3. When the New Group page appears, enter `DB2eAdjusters` in the Group Name field and then click **Create Group**.

Create users

To create users in the WebSphere Everyplace Deployment User Management Console, do the following:

1. Click the **Users** tab from the User Management Console.
2. Click the **Create New User** button.
3. When the New User page appears, enter the user ID, first name, last name, and password (as seen in Figure 13-3 on page 353), and then click **Create User**.

   For our example we created the users listed in Table 13-1 on page 350.


New User - Microsoft Internet Explorer

Address http://wedswin1.itso.ral.ibm.com/user-management/NewUserPage.do

Everyplace Deployment User / Group Management

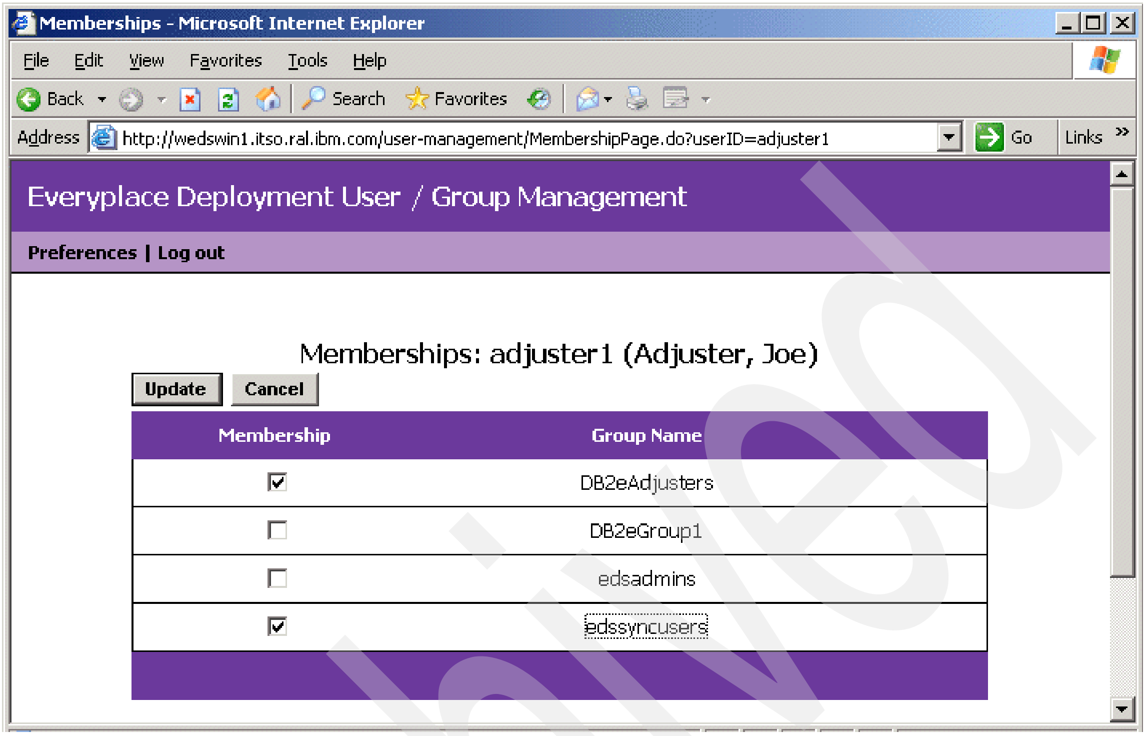
Preferences | Log out

New User

User ID:	adjuster1
First Name:	Joe
Last Name:	Adjuster
Password:	●●●●●●●●●
Confirm Password:	●●●●●●●●●

Figure 13-3 User Management Console - Create user

4. Click **Memberships** for the newly created user (for example, adjuster1).
5. When the Memberships page appears, do the following (as seen in Figure 13-4 on page 354), and then click **Update**:
   – Check **DB2eAdjusters**.
   – Check **edssyncusers**.



Memberships - Microsoft Internet Explorer

Address: http://wedswin1.itso.ral.ibm.com/user-management/MembershipPage.do?userID=adjuster1

Everyplace Deployment User / Group Management

Preferences | Log out

Memberships: adjuster1 (Adjuster, Joe)

Update | Cancel

Membership	Group Name
✓	DB2eAdjusters
☐	DB2eGroup1
☐	edsadmins
✓	edssyncusers

Figure 13-4 User Management Console - Membership

6. Repeat the process for each adjuster ID needed.
7. When done, click **Log out**.

13.2.3 Create the DB2e sync subscriptions

This section describes the steps to create the DB2e sync subscriptions for the ITSO example application.

The section is organized into the following tasks:

- Verify the status of the DB2e Sync Server.
- Refresh groups and users.
- Create a subscription set and assign to a group.
- Create a JDBC subscription.

Verify the status of the DB2e Sync Server

To verify the status of the DB2e Sync Server, do the following:

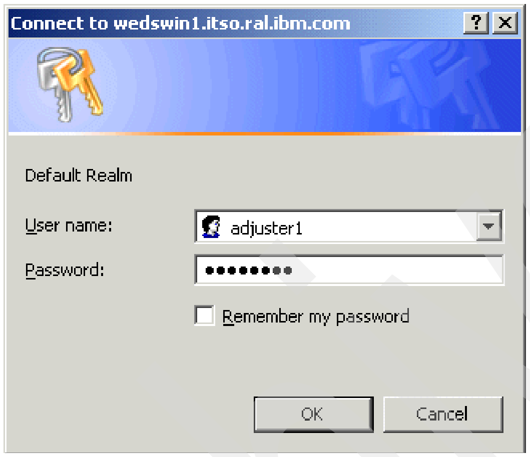
1. Ensure the DB2 Everyplace Server is started.

2. Enter the following URL in a Web browser:

```
http://wedswin1.itso.ral.ibm.com/db2e/db2erdb
```

   Where wedswin1.itso.ral.ibm.com is the host name of the DB2 Everyplace Sync Server.

3. When prompted to log in, enter the user ID (for example, adjuster1) and password for the user created in “Create users” on page 352 (as seen in Figure 13-5), and then click **OK**.



Connect to wedswin1.itso.ral.ibm.com

Default Realm

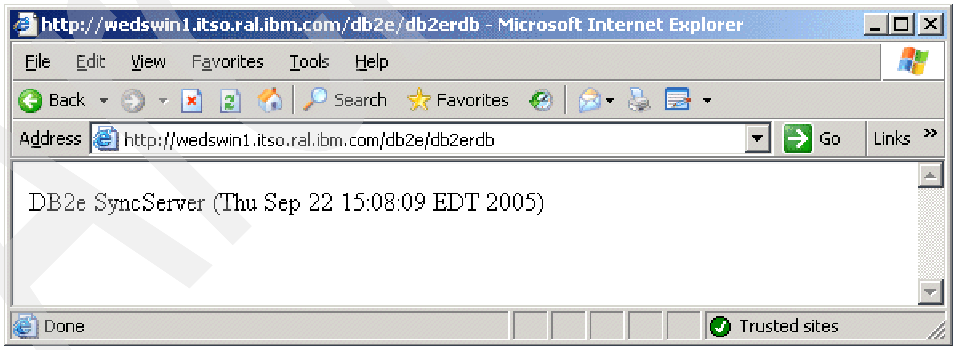
User name: adjuster1

Password: •••••••••

Remember my password

Figure 13-5 DB2 Everyplace Sync Server login

If successful, you will see a dialog like Figure 13-6.



http://wedswin1.itso.ral.ibm.com/db2e/db2erdb - Microsoft Internet Explorer

Address: http://wedswin1.itso.ral.ibm.com/db2e/db2erdb

```
DB2e SyncServer (Thu Sep 22 15:08:09 EDT 2005)
```

Done

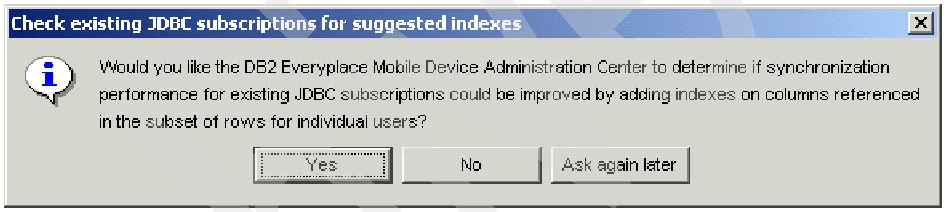
Trusted sites

Figure 13-6 DB2 Everyplace Sync Server login output

Refresh groups and users

To refresh the groups and users within the DB2e Mobile Device Administration Center (MDAC), do the following:

1. Start the Mobile Device Administration Center by clicking **Start** → **Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **DB2 Everyplace Deployment** → **DB2 Everyplace Sync** → **Start Mobile Device Administration Center**.
2. When the WPS LDAP Logon dialog appears, enter the following and click **OK**:
   - User name: wedadmin
   - Password: <password>
   - Sync group: edssyncusers
3. You may see the "Check existing JDBC subscriptions for suggested indexes" dialog, as seen in Figure 13-7. Click **No** (none needed).



Check existing JDBC subscriptions for suggested indexes

Would you like the DB2 Everyplace Mobile Device Administration Center to determine if synchronization performance for existing JDBC subscriptions could be improved by adding indexes on columns referenced in the subset of rows for individual users?

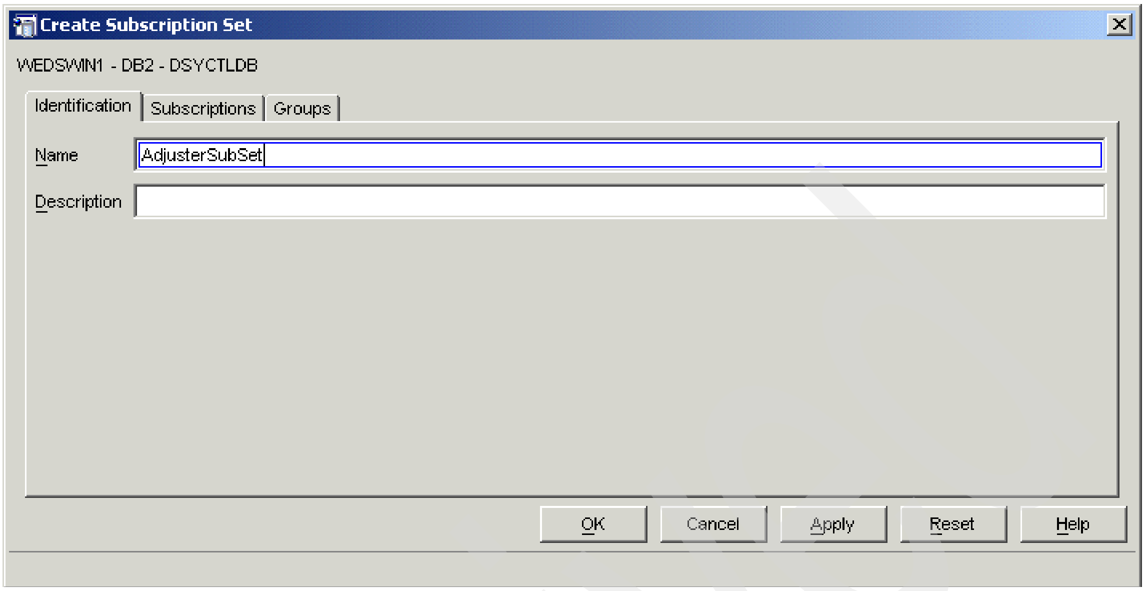
Figure 13-7 Check existing JDBC subscriptions for suggested indexes

4. Right-click **Groups** and select **Refresh WPS LDAP Groups**.
5. Right-click **Users** and select **Refresh WPS LDAP Users**.

Create a subscription set and assign to a group

To create a DB2 Everyplace subscription set and assign to a group, do the following from the Mobile Device Administration Center:

1. Right-click **Subscription sets** and select **Create**.
2. Click the **Identification** tab and enter `AdjusterSubSet` in the Name field, as seen in Figure 13-8 on page 357.


Create Subscription Set

WEDSWIN1 - DB2 - DSYCTLDB

Name: AdjusterSubSet

Description:

Figure 13-8 Create Subscription Set

3. Click the **Groups** tab.
4. Select the **DB2eAdjusters** group (created in “Create groups” on page 351) and click **>** to move it to the Selected groups list, as seen in Figure 13-9 on page 358, and then click **OK**.

   The AdjusterSubSet should be listed.

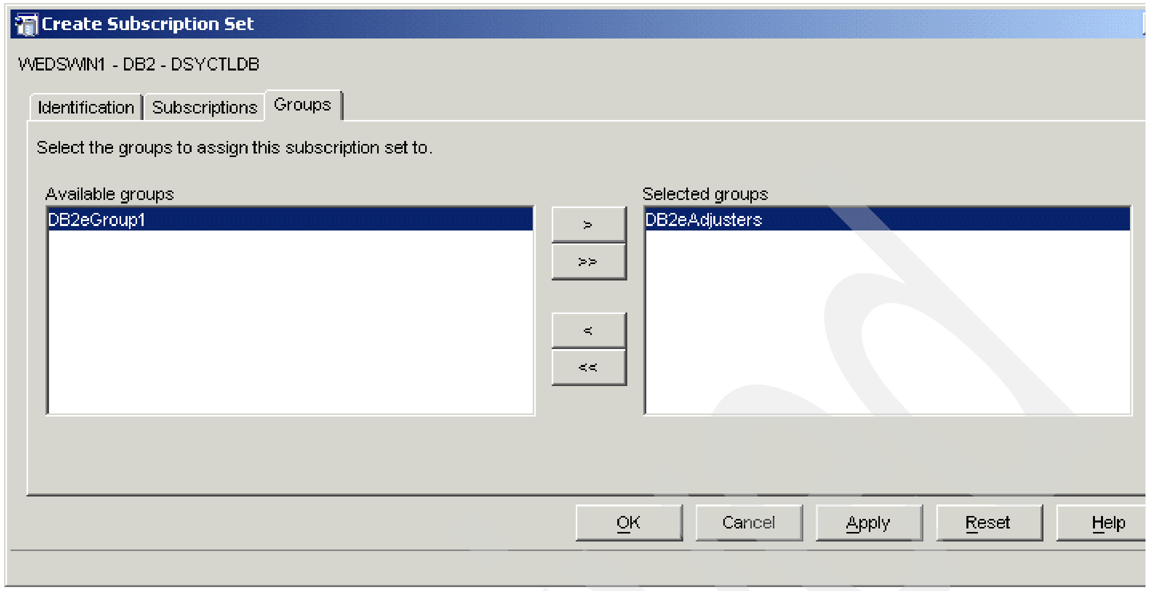
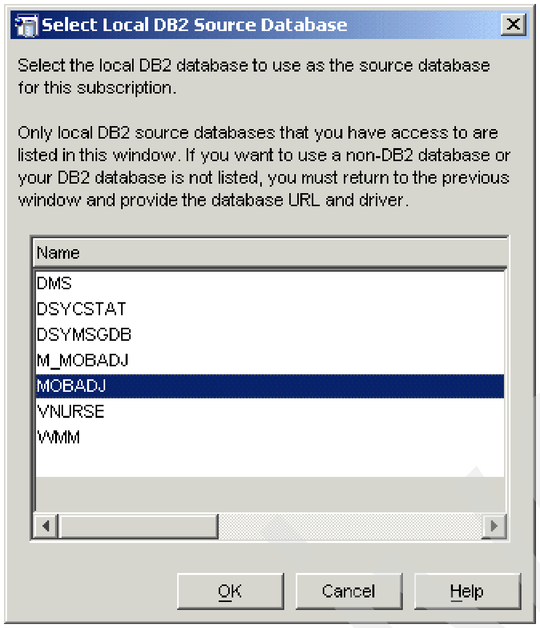


Figure 13-9 Create subscription set - Assign to group

Create a JDBC subscription

To create a DB2e JDBC subscription, do the following:

1. Right-click **Subscriptions** and select **Create** → **Table subscription** → **JDBC subscription**.
2. From the Identification tab, enter `AdjusterSub` in the Name field, and then select **DSYJDBC - Default DB2 Everyplace JDBC adapter** from the Adapter drop-down list.
3. Select the source database.
   a. Click the **Source** tab.
   b. To find the local DB2 application database, click the **...** button next to the Database URL field.
   c. When the Select Local DB2 Source Database dialog appears, select the **MOBADJ** (as seen in Figure 13-10 on page 359), and then click **OK**.


Select Local DB2 Source Database

Select the local DB2 database to use as the source database for this subscription.

Only local DB2 source databases that you have access to are listed in this window. If you want to use a non-DB2 database or your DB2 database is not listed, you must return to the previous window and provide the database URL and driver.

Name
DMS
DSYCSTAT
DSYMSGDB
M_MOBADJ
MOBADJ
VNURSE
WMM

Figure 13-10 Select source database

   d. Enter the user ID (for example, db2admin) and password for the DB2 administrator ID used to create the application databases in 13.2.1, "Create and populate the application database" on page 348.

   e. Click **Test connection**.

   f. You should see a message that the connection was successful. Click **Close**.

4. Select the mirror database.

   a. Click the **Mirror** tab.

   b. To find the local DB2 application database, click the **...** button next to the Database URL field.

   c. When the Select DB2 Mirror Database dialog appears, select the **M_MOBADJ** database, and then click **OK**.

   d. Enter the user ID (for example, db2admin) and password for the DB2 administrator ID used to create the application databases in 13.2.1, "Create and populate the application database" on page 348.

   e. Click **Test connection**.

   f. You should see a message that the connection was successful. Click **Close**.

5. Add a subscription set:
   a. Click the **Subscriptions sets** tab.
   b. Select **AdjustersSubSet** (created in “Create a subscription set and assign to a group” on page 356) and click the **>** to it to the Selected subscriptions list.
6. Define the subscription:
   a. Click the **Identification** tab. Note that now that the subscription set has been added, the Define subscription button is enabled.
   b. Click **Define subscription**.
   c. Click **Add** to add tables to the subscription.
   d. Select the **<user>.CLAIMS**, **<user>.CUSTOMERS**, **<user>.PARTS**, and **<user>.POLICIES** tables, as seen in Figure 13-11.

      Where <user> is the DB2 administror users specified when the application databases were created in 13.2.1, “Create and populate the application database” on page 348.

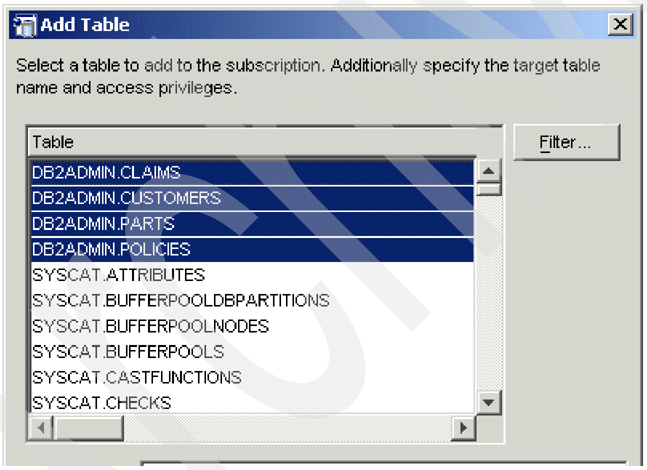

Select a table to add to the subscription. Additionally specify the target table name and access privileges.

Table
DB2ADMIN.CLAIMS
DB2ADMIN.CUSTOMERS
DB2ADMIN.PARTS
DB2ADMIN.POLICIES
SYSCAT.ATTRIBUTES
SYSCAT.BUFFERPOOLDBPARTITIONS
SYSCAT.BUFFERPOOLNODES
SYSCAT.BUFFERPOOLS
SYSCAT.CASTFUNCTIONS
SYSCAT.CHECKS

Figure 13-11 Select tables to add to subscription

   e. Select **MOBADJSP1** from the Source table space drop-down list.
   f. Select **USERSPACE1** from the Mirror table space drop-down list.
   g. Click **Add**.
   h. Click **Close**.
7. Set the subscription timing value.

By default, the subscription to synchronize the application database and DB2e mirror database is set to 3600 seconds. For our example scenario, we chose to set this to 120 seconds to facilitate the application walkthrough. You will need to consider the performance implications of this setting given the complexity of your database.

a. Click **Timing** on the Define Replication Subscription AdjusterSub dialog.
b. Enter the desired timing value in the Batch window and click **OK**; for example, we entered 120 (default is 3600 seconds).
c. When you return to the Define Replication Subscriptions dialog, click **OK**.
d. When the Create JDBC Subscription window appears, check **Replicate now** and then click **OK**.

   DB2 Everyplace will create internal replicas of the selected tables.

13.2.4 Install the MQe Claim Server application

This section describes how to install and run the MQe Claim Server for the ITSO example.

Install the MQe Claim Server

To install the MQe Claim Server application component for the ITSO example, do the following:

1. Create the C:\MQe_MOBADJ directory on the Access Server node.
2. Copy the C:\6775code\deploy\server\MobileAdjusterMQServer.jar and runMQeServer.bat to the C:\MQe_MOBADJ directory.
3. Copy the C:\IBM\WebSphere\MQe\Java\Jars\MQeBundle.jar to the C:\MQe_MOBADJ directory.

Run the MQe Claim Server

To run the MQe Claim Server, do the following:

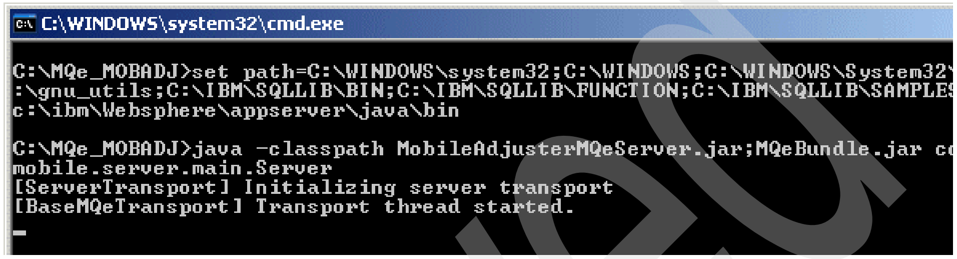
1. Open a command window and navigate to the C:\MQe_MOBADJ directory.
2. Set the Java environment. You can either add Java to the system path or define it manually at the command line, as follows:

```
set path=%path%;C:\IBM\WebSphere\AppServer\java\bin
```

Note: We recommend adding the above statement to the top of the runMQeServer.bat found in the C:\MQe_MOBADJ. Alternatively, add the Java environment to the system PATH environment variable.

3. Enter `runMQeServer.bat` to start the server. You should see something like Figure 13-12.

Note: The first time you start the MobileAdjuster MQ Everyplace Server you will see an exception because the queue manager does not exist; thus, it will create the queue manager.



```
C:\WINDOWS\system32\cmd.exe

C:\MQe_MOBADJ>set path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\
:\gnu_utils;C:\IBM\SQLLIB\BIN;C:\IBM\SQLLIB\FUNCTION;C:\IBM\SQLLIB\SAMPLES
c:\ibm\Websphere\appserver\java\bin

C:\MQe_MOBADJ>java -classpath MobileAdjusterMQeServer.jar;MQeBundle.jar co
mobile.server.main.Server
[ServerTransport] Initializing server transport
[BaseMQeTransport] Transport thread started.
_
```

Figure 13-12 MQ Everyplace Claim Server

Note: To stop the server, press Ctrl+C.

13.2.5 Install the Web application services

To install the Web application services for the ITSO example application, do the following:

- Launch the WebSphere Administrative Console.
- Create an authentication alias for the DB2 connection.
- Create the JDBC Provider for connecting to DB2.
- Create DataSource for connecting to the MOBADJ database.
- Set the DB2UNIVERSAL_JDBC_DRIVER_PATH variable.
- Test the connection to the MOBADJ database.

Launch the WebSphere Administrative Console

To access the WebSphere Application Server Administrative Console, do the following:

1. Ensure the WebSphere Everyplace Deployment Servers are started. For details refer to 13.1.3, “Server startup” on page 346.
2. Ensure the server1 application server is started.
   a. Open a command window and navigate to the following directory:

      `C:\IBM\WebSphere\AppServer\profiles\default\bin`

   b. Enter the following command to check the status of server1:

      `serverstatus server1`

   c. If server1 is not started, enter the following to start the server:

      `startserver server1`

3. Enter the following URL in a Web browser to access the WebSphere Administrative Console:

   `http://wedswin1.itso.ral.ibm.com:9060/ibm/console`

4. Enter a user ID and password (for example, wedadmin) and click **Log in**.

   > **Note:** WebSphere Security is now enabled on the node; thus, a user ID and password are required.

Create an authentication alias for the DB2 connection

To create an authentication alias for the DB2 connection, do the following from the WebSphere Administrative Console:

1. Expand **Security** and click **Global security**.
2. Expand **JAAS Configuration** under Authentication (right-hand side), and click **J2C Authentication data**.
3. Click **New**.
4. Enter the following and then click **OK**:
   – Alias: `MobileAdjuster`
   – User ID: `db2admin`
   – Password: `<password>`
   – Description: `Credentials for connecting to the MODADJ`

   > **Note:** This new user alias will be used by the MobileAdjuster application to connect to the MODADJ database.

5. Click **Save**, and then click **Save** again to save to master configuration.

Create the JDBC Provider for connecting to DB2

To create the JDBC Provider for connecting to DB2, do the following:

1. Expand **Resources** and click **JDBC Providers**.
2. Ensure the **Node** scope is selected (indicated by red arrow).

   If not, click **Browse Nodes** and select the node, and then click **OK**.
3. Click **New**.

4. When the new JDBC providers page appears, do the following and then click **Next**:
   – Select the database type: Select **DB2**.
   – Select the provider type: Select **DB2 Universal JDBC Driver Provider**.
   – Select the implementation type: Select **XA data source**.
5. Review the values and click **OK** to complete the creation of the JDBC Provider.
6. Click **Save**, and then click **Save** again to save to master configuration.

Create DataSource for connecting to the MOBADJ database

To create the DataSource for connecting to the MODADJ database, do the following:

1. Expand **Resources** and click **JDBC Providers**.
2. Click **DB2 Universal JDBC Driver Provider** (created in "Create the JDBC Provider for connecting to DB2" on page 363).
3. Click **Data sources** under Additional Properties.
4. Click **New**.
5. When the New data source page appears, do the following:
   a. Enter `MobileAdjuster DataSource` in the Name field.
   b. Enter `jdbc/mobileadjuster` in the JNDI name field.
   c. Uncheck **Use this Data Source in container managed persistence (CMP)**.
   d. Select **DB2 Universal data store helper** from the Data store helper classes drop-down list.
   e. Select **wedswin1Node01/MobileAdjuster** from the *Component-managed* authentication alias drop-down list (where wedswin1Node1 is the node name for the ITSO environment).
   f. Select **wedswin1Node01/MobileAdjuster** from the *Container-managed* authentication alias drop-down list (where wedswin1Node01 is the node name for the ITSO environment).
   g. Enter `MOBADJ` in the Database name field.
   h. Click **OK**.
6. Click **Save**, and then click **Save** again to save to master configuration.

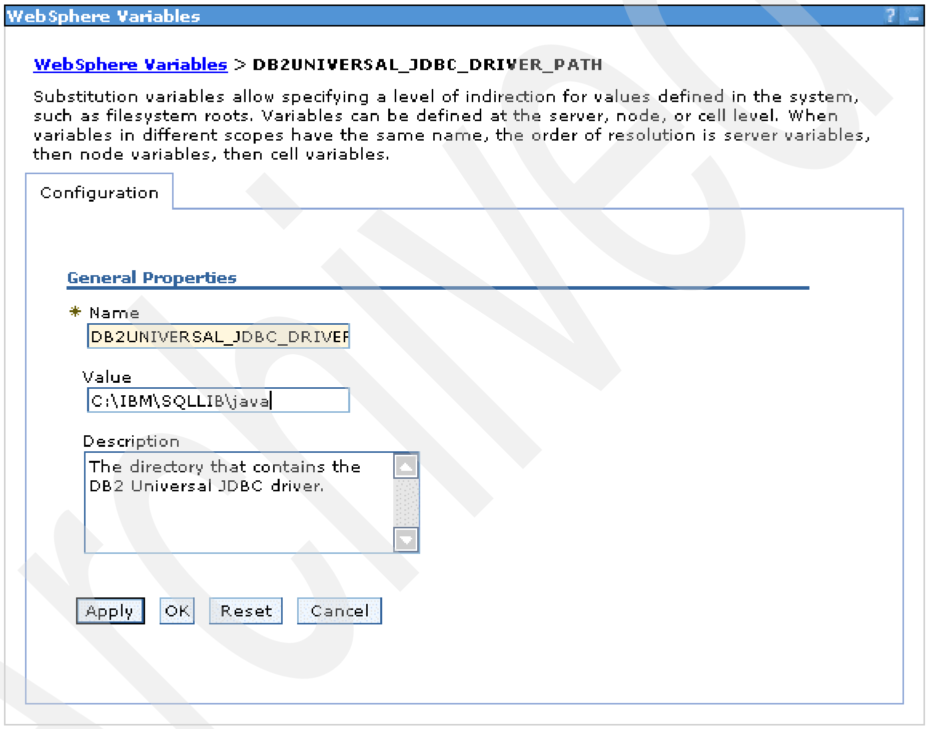
Set the DB2UNIVERSAL_JDBC_DRIVER_PATH variable

The JDBC Provider created in "Create the JDBC Provider for connecting to DB2" on page 363 includes a classpath variable to define the DB2 JDBC driver path.

This section describes how to update this variable based on your DB2 Universal Database installation directory.

To set the DB2UNIVERSAL_JDBC_DRIVER_PATH variable, do the following

1. Expand **Environment** and click **WebSphere Variables**.
2. Click **DB2UNIVERSAL_JDBC_DRIVER_PATH**.
3. Enter the DB2 Universal Database path (<db2_home>\java) to the db2jcc.jar file (based on your installation) in the Value field, as seen Figure 13-13, and then click **OK**.


WebSphere Variables > DB2UNIVERSAL_JDBC_DRIVER_PATH

Substitution variables allow specifying a level of indirection for values defined in the system, such as filesystem roots. Variables can be defined at the server, node, or cell level. When variables in different scopes have the same name, the order of resolution is server variables, then node variables, then cell variables.

Configuration

General Properties

* Name: DB2UNIVERSAL_JDBC_DRIVEF

Value: C:\IBM\SQLLIB\java

Description: The directory that contains the DB2 Universal JDBC driver.

Figure 13-13 Set DB2UNIVERSAL_JDBC_DRIVER_PATH variable

4. Click **Save**, and then click **Save** again to save to master configuration.

Test the connection to the MOBADJ database

To test the connection to the MOBADJ database, do the following:

1. We recommend that you restart the server1 application server to ensure the configuration settings take effect properly.
2. Expand **Resources** and click **JDBC Providers**.

3. Click **DB2 Universal JDBC Driver Provider** (created in “Create the JDBC Provider for connecting to DB2” on page 363).
4. Click **Data sources** under Additional Properties.
5. Check the **MobileAdjuster DataSource** check box.
6. Click **Test Connection**. You should see a message like the following:

```
Test connection for data source MobileAdjuster DataSource on server server1
at node wedswin1Node01 was successful.
```

13.2.6 Install the MobileAgent enterprise application

This section describes how to install and verify the MobileAgent, which is the online Web application used by agents to create insurance claims.

Install the MobileAgent

To install the MobileAgent servlet, do the following from the WebSphere Application Server Administrative Console:

1. Expand **Applications** and click **Install New Application**.
2. Select **Local file system**, and enter the path to the MobileAgent.ear (for example, C:\6775code\deploy\server\MobileAgent.ear) in the Specify path field, and then click **Next**.
3. When the Preparing for the application installation page appears, accept the default settings and click **Next**.
4. When the Select installation options (Step 1) page appears, accept the default settings and click **Next**.
5. When the Map modules to servers (Step 2) page appears, do the following:
   a. Select both **server1** and **webserver1** from the Clusters and Server list (use the Ctrl key to select multiple entries).
   b. Check the check box next to **MobileAgent**, and then click **Apply** (next to Clusters and Servers list).

      The server1 and webserver1 servers should be listed under the Server heading.
   c. Click **Next**.
6. When the Map resource references to resource (Step 3) page appears, do the following:
   a. Scroll down the page, and check the check box next to **MobileAgent**.
   b. Select **wedswin1Node01/MobileAdjuster** from the Select authentication data entry drop-down list, and then click **Apply**.

   c. Click **Next**.
   d. When the Application Resource Warning page appears, click **Continue**.
7. When the Map virtual hosts for Web modules (Step 4) page appears, click **Next**.
8. When the Summary (Step 5) page appears, click **Finish**.

   You should see a message like the following if successful:

   ```
   Application MobileAgent installed successfully.
   ```
9. Click **Save to Master Configuration**, and then click **Save**.

Verify the MobileAgent

To verify the MobileAgent, do the following from the WebSphere Application Server Administrative Console:

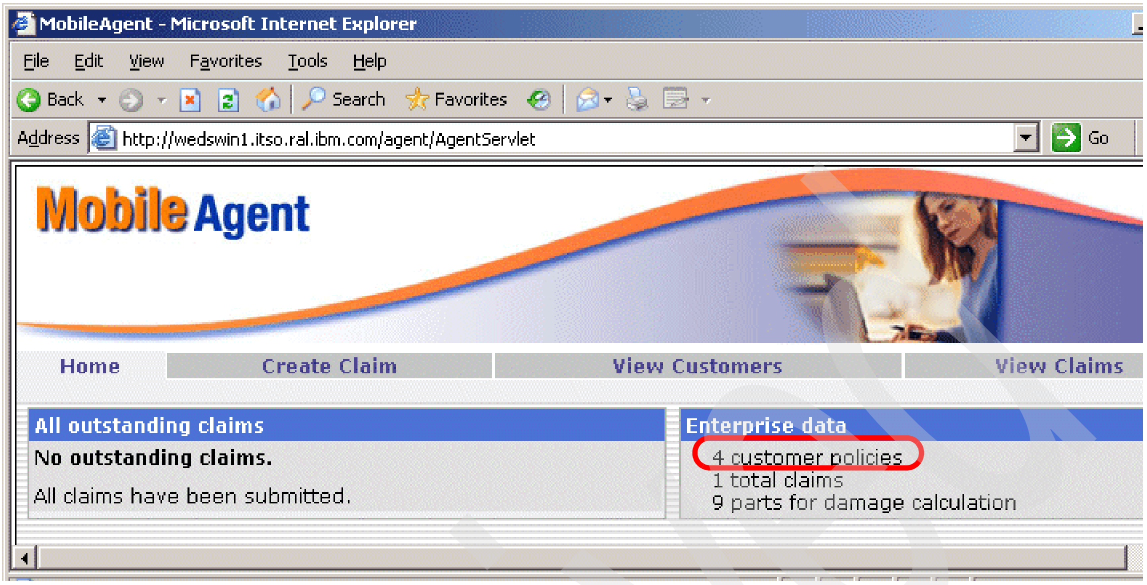
1. Expand **Applications** and click **Enterprise Applications**.
2. Check **MobileAgent** and click **Start**.

   You should see a green arrow to indicate the application is started.
3. Enter the following URL in a Web browser:

   ```
   http://wedswin1.itso.ral.ibm.com/agent/AgentServlet
   ```

   You should see a page like Figure 13-14 on page 368.

> **Note:** When the MOBADJ database was created, four customer policies were added in the sample data. You should see four customer policies listed in the MobileAgent page.
>
> This information can be used to verify that the AgentServlet is communicating properly with the database. If no data is retrieved/displayed, review your configuration settings to ensure you have configured your server properly.


MobileAgent - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://wedswin1.itso.ral.ibm.com/agent/AgentServlet Go
Mobile Agent
Home Create Claim View Customers View Claims

All outstanding claims

No outstanding claims.

All claims have been submitted.

Enterprise data

4 customer policies
1 total claims
9 parts for damage calculation

Figure 13-14 MobileAgent Web page

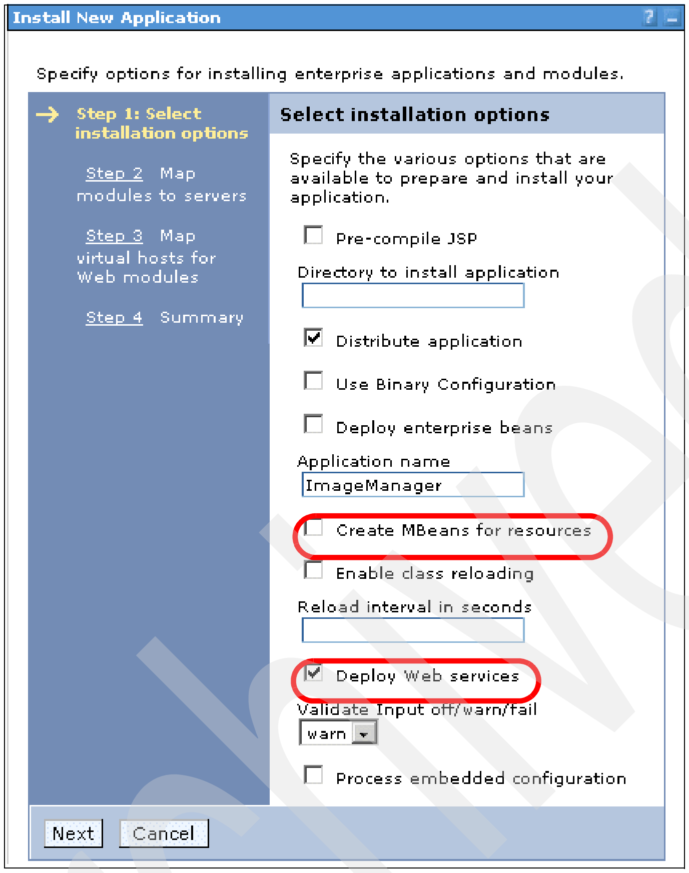
13.2.7 Install the Image Manager Web service

This section describes how to install and verify the Image Manager Web service for the MobileAdjuster application.

Install the Image Manager ear

To install the Image Manager ear file, do the following:

1. Log in to the WebSphere Administrative Console. For details refer to “Launch the WebSphere Administrative Console” on page 362.
2. Expand Applications and click **Install New Application**.
3. Select **Local file system**, enter `C:\6775code\deploy\server\ImageManager.ear` in the Specify path field, and then click **Next**.
4. When the Preparing for the application installation page appears, accept the default settings and click **Next**.
5. When the Select installation options (Step 1) page appears, do the following (as seen in Figure 13-15 on page 369), and then click **Next**:
   – Uncheck **Create Mbeans for resources**.
   – Check **Deploy Web services**.



Install New Application

Specify options for installing enterprise applications and modules.

→ Step 1: Select installation options

Step 2 Map modules to servers

Step 3 Map virtual hosts for Web modules

Step 4 Summary

Select installation options

Specify the various options that are available to prepare and install your application.

☐ Pre-compile JSP

Directory to install application

☑ Distribute application

☐ Use Binary Configuration

☐ Deploy enterprise beans

Application name: ImageManager

☐ Create MBeans for resources

☐ Enable class reloading

Reload interval in seconds

☑ Deploy Web services

Validate Input off/warn/fail: warn

☐ Process embedded configuration

Next | Cancel

Figure 13-15 Install Image Manager - Step 1: Select installation options

6. When the Map modules to servers (Step 2) page appears, do the following:
   a. Select both **server1** and **webserver1** from the Clusters and Servers list.
   b. Check the check box next to **ImageManager**, and then click **Apply** (next to Clusters and Servers list).

      The server1 and webserver1 servers should be listed under the Server heading.
   c. Click **Next**.
7. When the Map virtual hosts for Web modules (Step 3) page appears, accept the default settings and click **Next**.
8. When the Provide options to perform the Web Services deployment (Step 4) page appears, accept the default settings and click **Next**.
9. When the Summary (Step 5) page appears, click **Finish**.

You should see a message like the following if successful:

```
Application ImageManager installed successfully.
```

10. Click **Save to Master Configuration**, and then click **Save**.

Verify the Image Manager installation

To verify the Image Manager Web service installation, do the following from the WebSphere Administrative Console:

1. Expand **Applications** and click **Enterprise Applications**.
2. Check the check box next to **ImageManager** and then click **Start**.

   You should see a green arrow to indicate the application is started.

13.3 Client-side application deployment

This section describes how to install the client-side application components. There are two methods possible for installing application components on the client platform:

- Remote software distribution

  > **Note:** For details on installing the ITSO MobileAdjuster client-side application as a remote Device Manager server software distribution job, refer to 14.2, “Managing clients for the ITSO working example” on page 381.

- Local install of client-side application

  The remainder of this chapter describes how to install the client-side application on the local device.

13.3.1 Prerequisites for client-side application deployment

Before installing the client-side application, ensure the following WebSphere Everyplace Deployment servers are installed and running on the WED server node (in our example, wedswin1).

- Ensure the IBM HTTP Server Windows service is started.

  For details refer to “Server startup” on page 346.

- Ensure the following WebSphere Everyplace Deployment application servers are started:
  - coreServices
  - DMS_AppServer

  – IBMDB2eServer

  For details refer to “Server startup” on page 346.

- Ensure the server1 application server hosting the following ITSO MobileAdjuster example applications is started:
  – *MobileAgent* servlet
  – *ImageManager* Web service

  For details refer to “Server startup” on page 346.

- MQe Claim Processing Server

  For details refer to “Run the MQe Claim Server” on page 361.

13.3.2 Local install of client-side application

This section describes how to install the client-side application for the ITSO example application on the local Access Client node.

Note: This procedure requires that you have unpacked the 6775code.zip on the Access Client node (refer to “Download the ITSO sample code zip” on page 346).

Install the MobileAdjuster application

To install the MobileAdjuster application, do the following on the Access Client node:

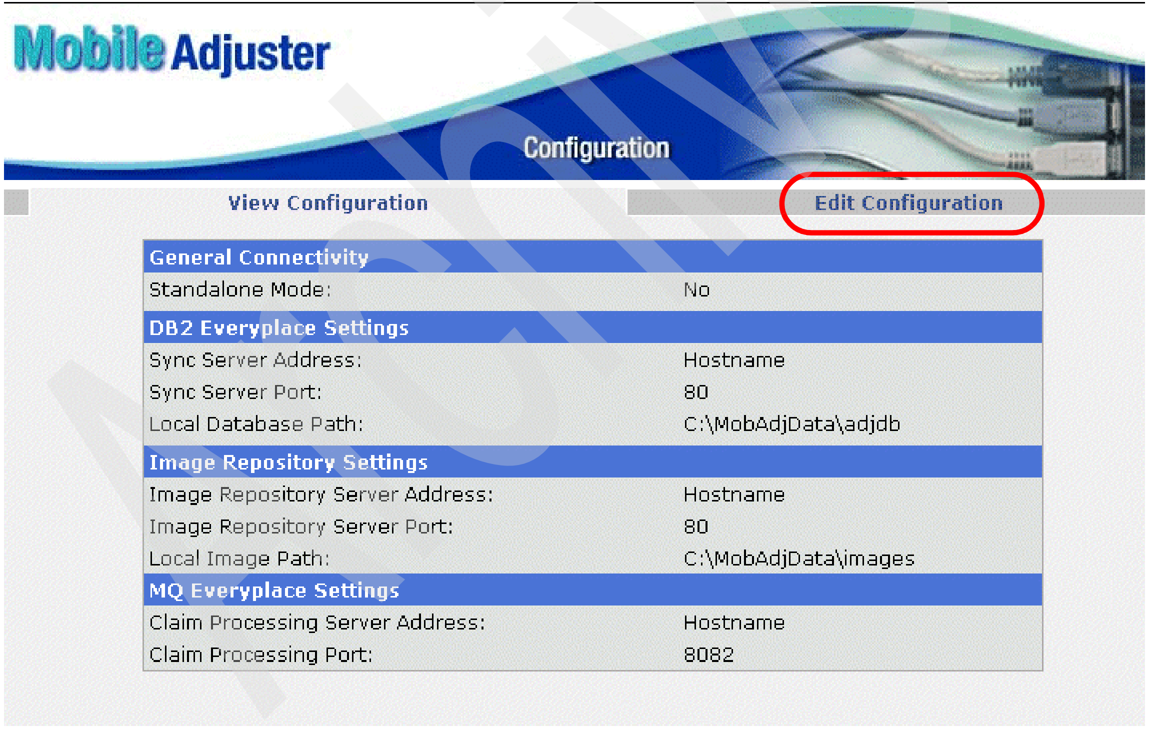
1. Start the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux by clicking **Start** → **Programs** → **IBM WebSphere Everyplace Deployment** → **WebSphere Everyplace Deployment 6.0.0**.
2. Select **Application** → **Install**.
3. Click **Add Zip/Jar Location**, enter `C:\6775code\deploy\client\MobileAdjuster-UpdateSite.zip`, and then click **OK** to accept the default label.
4. Check **MobileAdjuster-UpdateSite.zip** from the Locations list (uncheck others if checked), and then click **Next**.
5. Check **MobileAdjusterFeature** from the Features list and then click **Next**.
6. When the Feature License agreement appears, review the terms and, if in agreement, select **I accept the terms in the license agreement**, and then click **Next**.
7. When the Install Location dialog appears, accept the default settings and click **Finish**.
8. When the Feature verification warning dialog appears, click **Install**.

9. When prompted to restart the workbench for changes to take effect, click **Yes**.
10. Select **Applications** → **Open** from the WebSphere Everyplace Deployment menu to verify the following three applications exist:
    - MobileAdjuster Config
    - MobileAdjuster Images
    - MobileAdjuster

Configure the MobileAdjuster application

To configure the MobileAdjuster application, do the following:

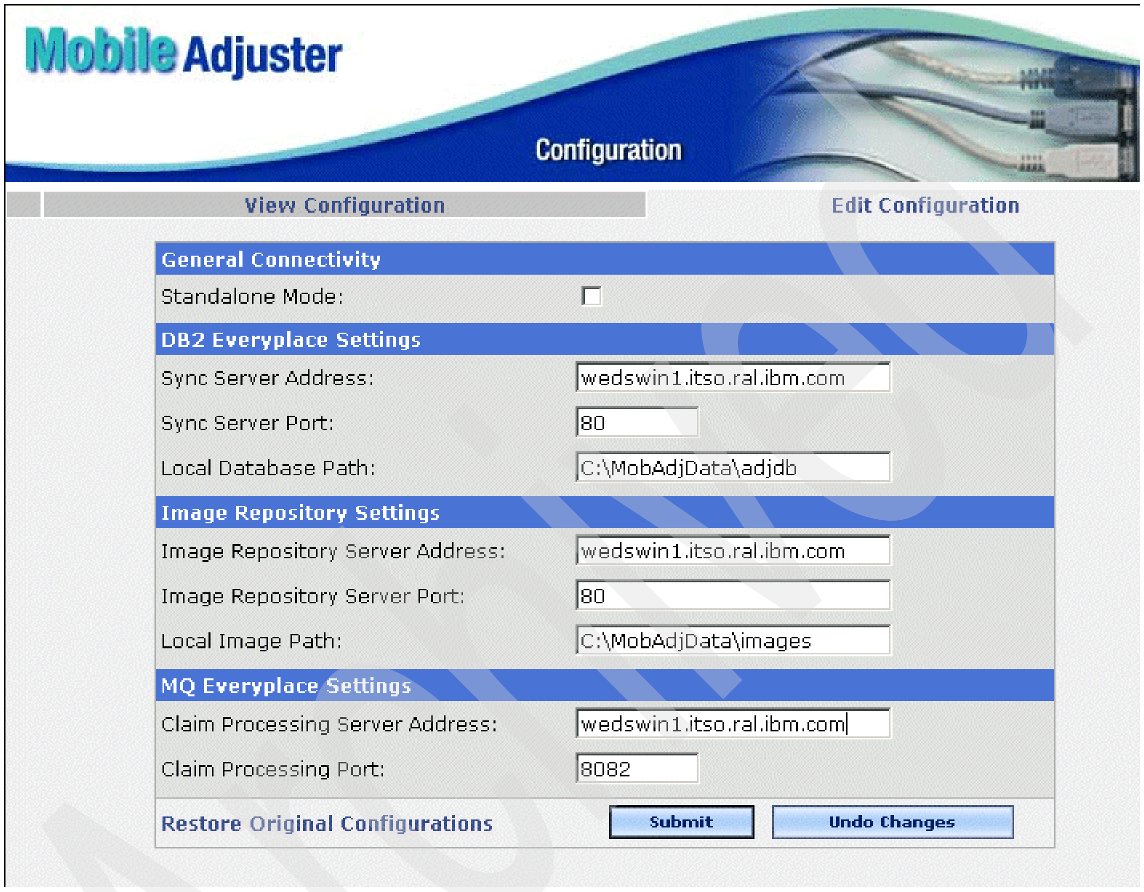
1. Start the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux on the Access Client node.
2. Click **Open** → **Mobile Adjuster Config**.
3. When the Mobile Adjuster Configuration page appears, click **Edit Configuration**, as seen in Figure 13-16.



General Connectivity	
Standalone Mode:	No
DB2 Everyplace Settings	
Sync Server Address:	Hostname
Sync Server Port:	80
Local Database Path:	C:\MobAdjData\adjdb
Image Repository Settings	
Image Repository Server Address:	Hostname
Image Repository Server Port:	80
Local Image Path:	C:\MobAdjData\images
MQ Everyplace Settings	
Claim Processing Server Address:	Hostname
Claim Processing Port:	8082

Figure 13-16 Mobile Adjuster - Edit Configuration

4. Enter your configuration preferences. For the ITSO example runtime environment, we entered the values displayed in Figure 13-17, and then clicked **Submit**.



General Connectivity

Standalone Mode:	☐

DB2 Everyplace Settings

Sync Server Address:	wedswin1.itso.ral.ibm.com
Sync Server Port:	80
Local Database Path:	C:\MobAdjData\adjdb

Image Repository Settings

Image Repository Server Address:	wedswin1.itso.ral.ibm.com
Image Repository Server Port:	80
Local Image Path:	C:\MobAdjData\images

MQ Everyplace Settings

Claim Processing Server Address:	wedswin1.itso.ral.ibm.com
Claim Processing Port:	8082

Figure 13-17 Mobile Adjuster - Configuration settings

Verify the MobileAdjuster application

To verify the MobileAdjuster application, do the following:

1. Start the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux on the Access Client node.
2. Click **Open** → **Mobile Adjuster**.
3. When the Mobile Adjuster Log in page appears, enter the adjuster user ID and password created in “Create users and groups” on page 350 (for example, adjuster1), and then click **Login**.

The first time you log in there will be no local claims or customer data on the client. The Store data will display zeros for policies, claims, and damages until you refresh the data.

4. Click **Refresh Data**.

   After the refresh is complete, you should see the stored data values change.

For a full application walkthrough, refer to Chapter 15, “Application walkthrough” on page 407.

14

Managing clients

Now that the disconnectable application is built, tested, and ready to be used by the mobile workforce, a convenient method needs to be available to install the application on the many clients (laptops and devices). In addition, as updates to the application are made, it is essential to know that the clients have been updated with the latest version. To further extend the Access Integration patterns established in the previous chapters, we now shift our focus to server-based client management.

Client management allows an enterprise to perform the following actions:

- Device configuration: Setting device parameters for hardware or software
- Inventory collection: Collecting hardware and software information about the device
- Software distribution: Distributing, installing, and removing software or data files for the device
- Initial provisioning: Providing initial access for devices to the Device Manager server and restoring the Device Manager configuration information

14.1 Managing clients overview

IBM Tivoli Device Manager Server (DMS) component, also known as Device Manager, is the IBM client management software solution. DMS helps enterprises manage many different types of clients. These client devices include desktops, laptops, PDAs, and Smartphones. Clients are enrolled into a database where many different management tasks can be performed on the device.

DMS is included in the following IBM mobile related software products:

- IBM WebSphere Everyplace Deployment (WED)
- IBM WebSphere Everyplace Access (WEA)
- IBM Tivoli Provisioning Manager (TPS)
- IBM WebSphere Everyplace Device Manager (WEDM)

The remaining sections of the managing clients overview are organized as follows:

- WebSphere Everyplace Deployment supported clients
- Device Manager server and client components
- Server-to-device communication
- Device enrollment
- Device management jobs
- Device management administration

14.1.1 WebSphere Everyplace Deployment supported clients

Our focus in this book is on the IBM WebSphere Everyplace Deployment V6.0 (WED) product. WED supports the following clients that include OSGI device agents:

- IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux

  This client platform is targeted at laptop and desktop clients.

- IBM Workplace Client Technology, Micro Edition V5.7.1, 5.7.2 (WCTME)

  This client platform is targeted at PDA type devices such as the PocketPC and Palm.

- IBM Workplace Client Technology, Micro Edition Enterprise Offering V5.8.1 (WCTME-EO)

  This client platform is targeted at laptop and desktop clients, and is essentially replaced by IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux.

14.1.2 Device Manager server and client components

Figure 14-1 depicts the Device Manager server and client components.

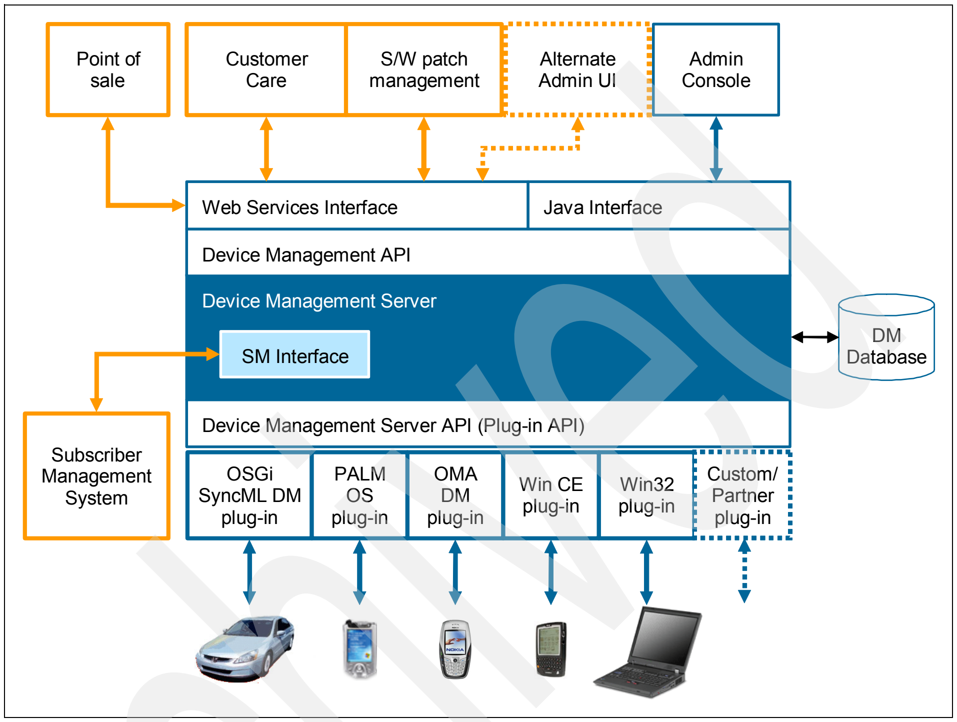


Figure 14-1 Device Manager components

From a server perspective, DMS is composed of four major components:

- Device Manager server (DMS)

  Manages and executes device management jobs variously in response to triggers. These triggers could include a device connecting to the server, a call to an API, or an administrator action.

- Device plug-ins

  Provide the logic that handles device identification, communications, job processing, and other high-level management tasks.

- Device Manager database

  Repository for all device management information. This includes device inventory information, device management jobs, package meta data, and many other details.

- Device Management API

Programming interface between the Device Manager server, administration clients, or external applications (Web Services and Java)

On the device side, a device agent is installed and communicates with the Device Manager server. Within IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux, the device agent is known as the *Enterprise Management Agent*. In the WED environment, the agent is triggered automatically by a polling interval set by either the user or the server administrator.

14.1.3 Server-to-device communication

The device agent communicates with the Device Manager server (DMS) over HTTP or HTTPS. The device agent utilizes either a proprietary protocol, such as protocols included with the Microsoft PocketPC and Palm OS devices, or a industry standard protocol, such as Open Mobile Alliance - Device Management (OMA DM). The OMA DM protocol is used by the OSGi device agents as well as the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux.

Communication between the client and the Device Manager server can use either HTTP or HTTPS; therefore, requests and responses can pass through various network elements, such as firewalls.

14.1.4 Device enrollment

Device enrollment occurs when the device first becomes known to the device management system. This enrollment can be manual or programmatic. Manual enrollment takes place when an administrator or customer service representative creates a new device entry. It can be done programmatically by a workflow process.

Devices may enroll themselves upon their first connection to the Device Manager server under the following conditions:

- Device provides valid subscriber ID and credentials
- System is configured to allow devices to enroll themselves

At the time of first the connection, a set of jobs can be configured to be automatically processed for these new devices. Criteria is dynamically evaluated for each enrollment job to determine if it should be run for the new device.

14.1.5 Device management jobs

Device management actions and tasks are called jobs. The device management jobs can be initiated through the Device Management *Adminstrative Console* or

Administrative APIs. The jobs can be targeted to a single device, or group of devices. The primary job types are as follows:

- Inventory collection
- Device configuration
- Software distribution

Jobs can be scheduled to occur in the future, with the ability to set expiration times and retry capability in the event of a failure.

Jobs targeted to a device are run when the device connects to the Device Manager server.

Groups of devices can be characterized by the device owner, owner group, attributes of the device inventory, or a combination of the characteristics mentioned.

A history of jobs is maintained on the server and an administrator is able to tell whether a job has been delivered to the client and executed. Job success/failure is also tracked on a per-device basis.

In addition to the job types previously listed, other job types are as follows:

- Registry editing
- Registry retrieval
- Node discovery
- Bundle control
- Command script
- Custom command

Inventory collection job

For a device, inventory is typically collected for hardware, software, and configuration of the device. Information is specific to each device class and can include items such as computer model, installed cards, memory size, battery type, etc.

An inventory collection job obtains the inventory information from the device agent, returns the information to the Device Manager server, and stores the inventory information in the appropriate Device Manager database.

Inventory can be used for several potential applications:

- Asset and state tracking
  - Automated customized scans can be scheduled.
  - Raw inventory results can be used for report creation or automated job submission.

- Job device/target filtering
  - Any field data available via inventory can be used for mass device and job targeting.
  - Example filters include the Operating System name or installed software.

Device configuration job

Every device has configuration parameters that are specific to that device.

Configuration parameters identify:

- Device and device user (such as a device name and user ID)
- Network connection (such as phone number, DNS, and IP addresses)
- Services (such as printer address, POP server address, time server address, and others)

When a device enrolls, the administrator can set the configuration parameters for that device to initial values with a device configuration job. Likewise, an administrator can also change the configuration parameters at a later date with another device configuration job.

Software distribution jobs

Software distribution jobs send software to targeted devices. This software can consist of files, images, databases, installation processes, and native software.

Software Distribution supports two package formats:

- Proprietary format meta file
- OSGi standard bundle format

After a software package or OSGi bundle is registered with Device Manager and copied to the application server, the files that comprise the software cannot be modified on the application server. If any files are changed, the software registration must be removed and registered again.

For OSGi devices, such as Windows 32-bit and Linux devices running the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux, software is distributed as OSGi bundles.

An OSGi bundle is distributed as a JAR file and is comprised of Java classes and other resources, which together can provide functions to device owners and provide services and packages to other bundles.

14.1.6 Device management administration

Device management administration can be performed using the Device Manager Administration Console or using the API.

Device Manager Administration Console

An administrator can use the Device Manger Administration Console to manage devices by submitting various types of jobs to run on the devices and monitor the job progress.

Device Manager Administration API

The Device Manager Administration API is a set of Java classes used to manage devices, jobs, software, and queries. It is implemented as a set of Web services, which communicate using SOAP over HTTP. It is used to develop applications shipped with Device Manager. External applications not shipped with Device Manager can also use the Administration API.

14.2 Managing clients for the ITSO working example

The objective of this section is to demonstrate some key client management functions provided by the Device Manager server included with WED for the ITSO working example.

This section is organized into the following tasks:

- Environment prerequisites.
- Device Manager Console installation.
- WED Client installation.
- Device enrollment and inventory collection.
- Create a custom command job (enable local install).
- Software distribution.
- Device configuration.

14.2.1 Environment prerequisites

The scenarios in this section require that the Access Server node (WED Server) is installed and that the server-side application is deployed.

For details refer to the following:

- Section 12.2, “Access Server node installation” on page 329
- Chapter 13, “Application deployment” on page 345

14.2.2 Device Manager Console installation

The Device Manager Console is a Java-based (J2SE) graphical user interface (GUI) for administering device management operations.

As part of the WebSphere Everyplace Deployment installation, the Device Manager Console (DMconsole.zip) is installed in the <ihs_home>\htdocs\<locale>\dms\console directory. In order to run the DM Console, you must unpack the zip and configure the environment to run the DM Console.

There are two possibilities to consider for where you run the Device Manager Console:

- Run the Device Manager Console on the WED Server node.
- Run the Device Manager Console on a remote node.

Note: In either case, the Device Manager Console requires that DB2 Universal Database Client (or server) is installed. This is needed to access with the DMS database.

Install and configure the Device Manager Console

To install and configure the Device Manager Console, do the following:

1. Create the C:\IBM_DM directory.
2. Copy the <IHS_home>\htdocs\en_US\dms\console\DMconsole.zip to the C:\IBM_DM directory.
3. Unzip the DMconsole.zip in the C:\IBM_DM directory.
4. Create a shortcut to Device Manager Console for quick access to launch.
   a. From the Explorer, open the C:\IBM DM folder, right-click **DMconsole.bat**, and select **Create Shortcut** to place on your desktop.
   b. Rename the shortcut `Device Manager Console`.
5. Install IBM DB2 Universal Database V8.2 Client with Fixpack 9a.

   Note: We recommend that the DB2 UDB Client and Server have matching Fixpack levels.

   If this is the WED Server node, DB2 Universal Database Server can be used.

6. Configure the DB2 Universal Database Client.

Note: The following DB2 Universal Database client-server configuration is only needed when the Device Manager Console is configured on a remote node to the WED Server. If on the WED Server node, DB2 UDB is already installed and configured.

To configure the DB2 Universal Database Client to access the DB2 Universal Database Server, do the following:

a. Catalog the TCPIP node by entering the following in a DB2 command window:

```
db2 catalog tcpip node wedswin1 remote wedswin1.itso.ral.ibm.com server
50000
```

Syntax:

```
db2 catalog tcpip node <node_name> remote <db_server_name> server
<db2_service_name>
```

b. Verify the catalog of the TCPIP node by entering the following attach command in a DB2 command window:

```
db2 attach to wedswin1 user db2admin using <password>
```

Syntax:

```
db2 attach to <node_name> user <user_id> using <password>
```

Where <user_id> and <password> are those of the DB2 instance owner.

c. Catalog the DMS database by entering the following in a DB2 command window:

```
db2 catalog db dms at node wedswin1
```

Syntax:

```
db2 catalog db <db_name> at node <node_name>
```

Where <db_name> is the database on the remote DB2 server machine.

d. Verify the connection to the remote DMS database by entering the following in a DB2 command window:

```
db2 connect to dms user db2admin using <password>
```

Syntax:

```
db2 catalog db <db_name> at node <node_name>
```

Where <db_name> is the database on the remote DB2 server machine.

7. Copy <db2_home>\java\db2java.zip to the C:\IBM_DM\prereqs directory (created during unpack of DMconsole.zip).

Run and verify the Device Manager Console

To run the Device Manager Console, do the following:

1. Ensure the Device Manager server is started.
2. Double-click the **Device Manager Console** shortcut.

   Alternatively, run DMconsole.bat from the C:\IBM_DM directory.
3. When the Login Panel appears, enter the following, as seen in Figure 14-2, and then click **Login**:
   - User ID: `wedadmin`
   - Password: `<wedadmin_password>`
   - Device Manager server: `wedswin1.itso.ral.ibm.com`

Note: In our example, wedswin1.itso.ral.ibm.com is the host name of the WED Server where the DMS is running.

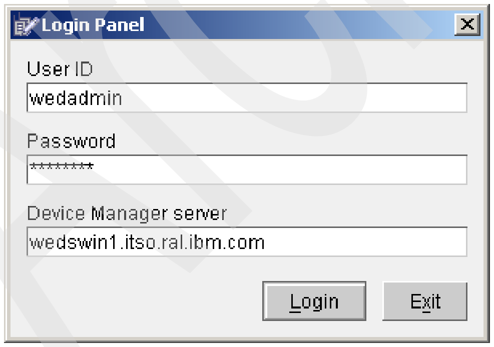


Figure 14-2 Device Manager Console logon

The Device Manager Console should appear, as seen in Figure 14-3 on page 385.

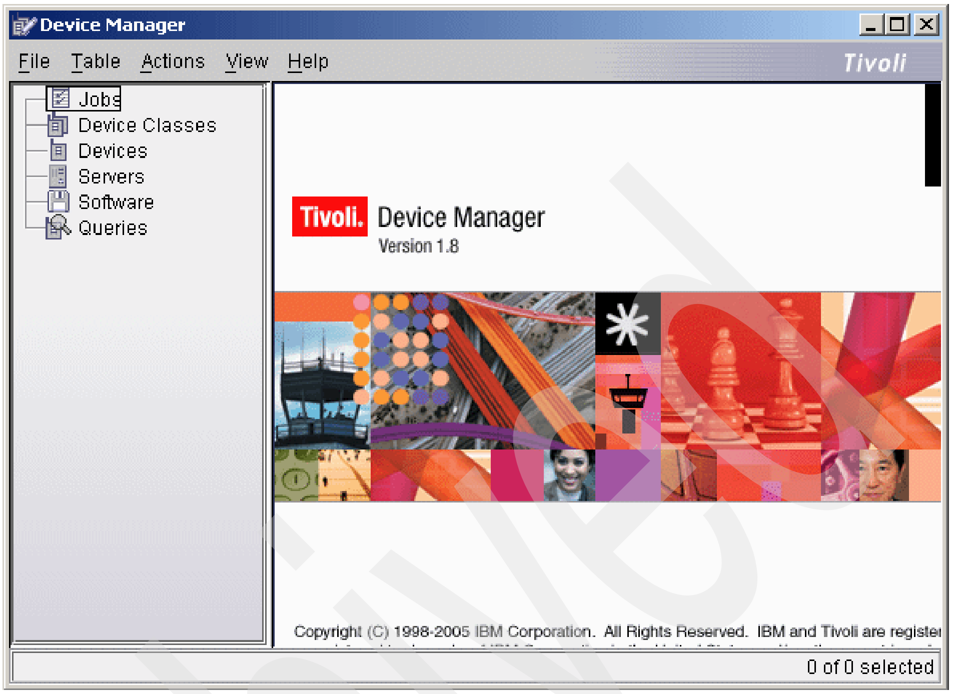

Device Manager
File Table Actions View Help
Tivoli
Jobs
Device Classes
Devices
Servers
Software
Queries
Tivoli. Device Manager
Version 1.8
Copyright (C) 1998-2005 IBM Corporation. All Rights Reserved. IBM and Tivoli are registe
0 of 0 selected

Figure 14-3 Device Manager Console

14.2.3 WED Client installation

The administrator has been given the task to ensure that all ITSO Insurance Adjusters have the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux installed on their laptops.

The IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux installation must be initiated from the client device. There are a couple of options possible for the type of installation:

- CD installation

> **Note:** For details on installing the client from a CD (or network file share) for the ITSO working example refer to 12.3, “Access Client node installation” on page 341.

- Network file share installation
- Web download site (appletInstaller - IBM technical preview)

The Web download site installation requires the use of the *appletInstaller,* which is only supported by IBM as a technical preview.

Note: For more information refer to the section “Setup a Web download site” in the *System Administrator’s Guide, IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux* product guide.

14.2.4 Device enrollment and inventory collection

As part of the ITSO Insurance working example, we need to prepare the WED client environment to deploy the client-side Mobile Adjuster application and manage the client. In this case, the adjuster has a Windows XP-based laptop with the IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux installed. Once the WED client is installed, we need to enroll the device with the Device Manager server. As part of this process, we also run an inventory collection job for the device and store the inventory information in the DMS.

This section describes how to perform the following tasks:

- Verify users and groups.
- Configure the Enterprise Management Agent.
- Inventory collection.
- Verify the enrollment and inventory collection job.

Verify users and groups

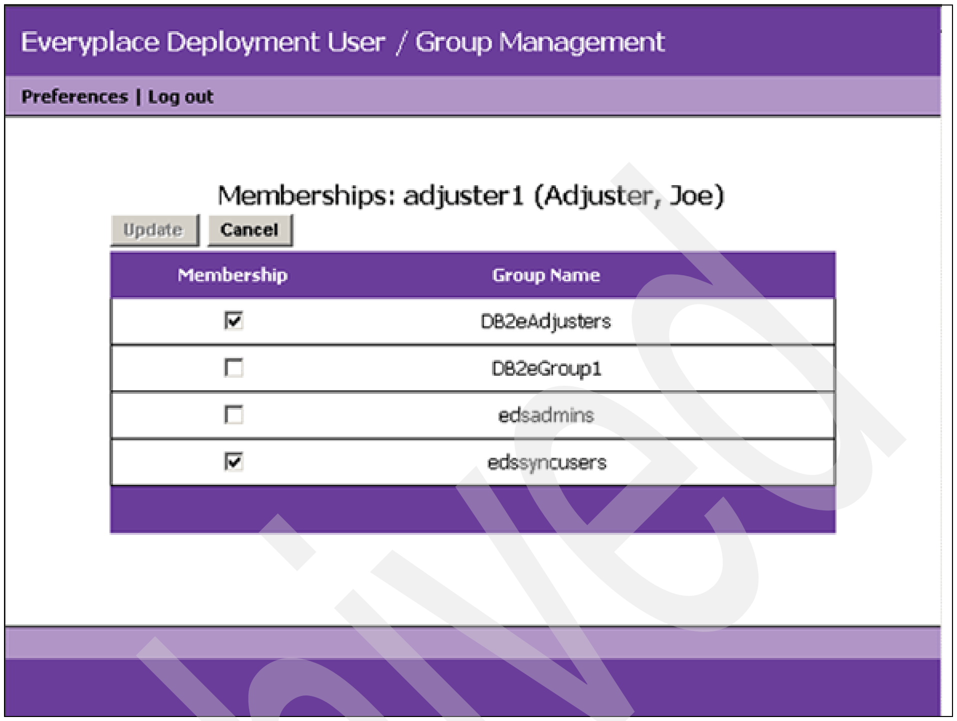
In our example, we created WebSphere Everyplace Deployment users and groups in the User Management Console as part of the server-side application deployment in 13.2.2, “Create users and groups” on page 350. We use the DB2eAdjuster group for management purposes.

Ensure that the Adjuster user IDs are added to the DB2eAdjuster group:

1. Enter the following URL in a Web browser to access the WED User Management Console:

   ```
   http://wedswin1.itso.ral.ibm.com/user-management/login.jsp
   ```

2. Log on with the WED admin user ID and password (for example, wedadmin).
3. Ensure the adjuster user IDs are part of the DB2eAdjusters group, as seen in Figure 14-4 on page 387.
4. When done, click **Log out**.


Everyplace Deployment User / Group Management

Preferences | Log out

Memberships: adjuster1 (Adjuster, Joe)

Update Cancel

Membership	Group Name
☑	DB2eAdjusters
☐	DB2eGroup1
☐	edsadmins
☑	edssyncusers

Figure 14-4 Group Membership window

Configure the Enterprise Management Agent

There are two options for configuring the Enterprise Management Agent on the WED Client to interact with the Device Manager server. First, the Enterprise Management Agent can be configured as part of the WED Client installation. Alternatively, the Enterprise Management Agent can be configured after the WED client installation. The information provided is the same for both cases.

To configure the Enterprise Management Agent on the WED Client after installation, do the following:

1. Ensure the WebSphere Everyplace Deployment servers are started.
2. Launch IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux.
3. Select **File** → **Preferences** from the WebSphere Everyplace Deployment menu.
4. Select **Enterprise Management Agent**.
5. When the Enterprise Management Agent dialog appears, do the following
   a. Check the **Enable Enterprise Management Agent** check box.

Note: Once the Enterprise Management Agent is enabled you will not be able to install, update, or uninstall features from the client (options disabled). These tasks will need to be managed from the DMS.

It is possible to re-enable the local install options from a server job (see "Create a custom command job (enable local install)" on page 393).

b. When the Warning dialog appears, as seen in Figure 14-5, click **Yes** for the ITSO working example.

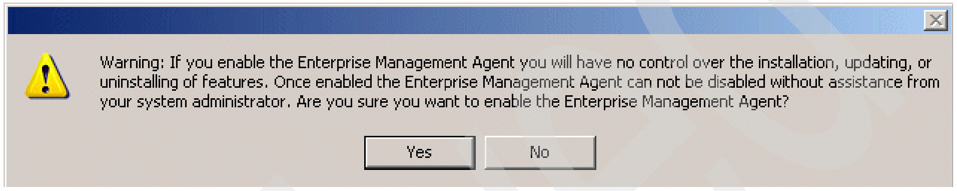

Warning: If you enable the Enterprise Management Agent you will have no control over the installation, updating, or uninstalling of features. Once enabled the Enterprise Management Agent can not be disabled without assistance from your system administrator. Are you sure you want to enable the Enterprise Management Agent?

Figure 14-5 Enterprise Management Agent warning

c. Enter `adjuster1` in the Device User Name field.

This user already exists (created in WED User Management Console).

d. Enter <password> in the Device User Password field.

e. Reenter the Device User Password.

f. Select **HTTP**.

g. Enter `<server_IP_address>/dmserver/SyncMLDMServletAuthRequired` in the Server IP Address field; for example:

```
wedswin1.itso.ral.ibm.com/dmserver/SyncMLDMServletAuthRequired
```

Where wedswin1.itso.ral.ibm.com is the host name of the WED Server. The IP address can also be used in place of the host name.

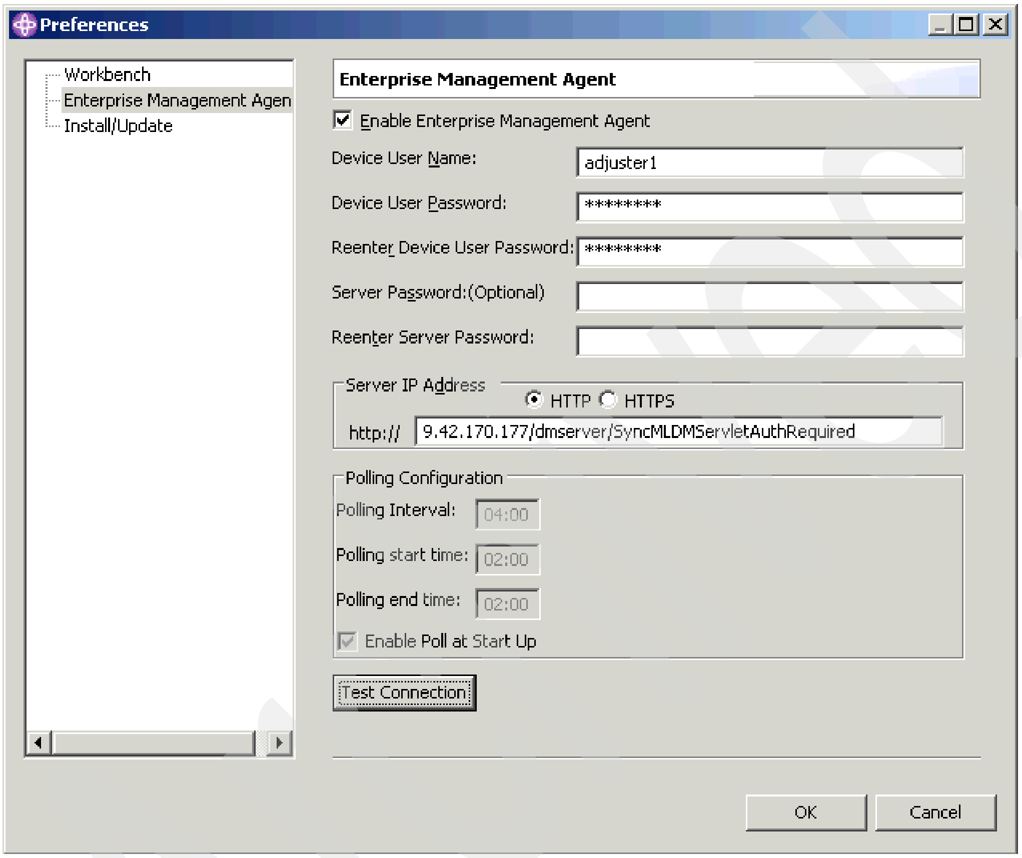
h. The Enterprise Management Agent dialog should look like Figure 14-6 on page 389. Click **Test Connection** to confirm the correct configuration.

i. You should see the message `Connection Successful`. Click **OK**.

Note: The Test Connection only verifies that the server entered is accessible. It does not verify that the servlet path has been entered correctly, or that the agent will successfully connect to the server.

j. Click **OK** on the Enterprise Management Agent dialog to save the settings.

Note: After clicking OK, you will not be able to install, update, or uninstall features on the client.


Preferences

Workbench
Enterprise Management Agen
Install/Update

Enterprise Management Agent

[✓] Enable Enterprise Management Agent

Device User Name: adjuster1

Device User Password: *********

Reenter Device User Password: *********

Server Password:(Optional):

Reenter Server Password:

Server IP Address [● HTTP] ○ HTTPS

http:// 9.42.170.177/dmserver/SyncMLDMServletAuthRequired

Polling Configuration

Polling Interval: 04:00

Polling start time: 02:00

Polling end time: 02:00

[✓] Enable Poll at Start Up

Figure 14-6 Configure the Enterprise Management Agent

Once the user chooses to enable the Enterprise Management Agent, DMS will have exclusive control over the WED client. This means new applications can only be installed through the Device Manager and no longer locally by the user. As a result, the Application → Install and Manage → Application → Application Management dialogs will be disabled on the WED client.

Important: Shortly after the user configures Enterprise Management Agent, the client will poll the Device Manager server automatically to enroll. This process may take a few minutes to complete.

For testing purposes, if you want to expediate this process, restart the WED client.

Inventory collection

Since the user device is a member of the Win32® device class, this job is executed at the time of enrollment. Please note that performing an inventory collection job collects a large amount of device-specific parameter data and may take several minutes to complete, particularly if the network connection for the device is slow. Please do not close the WED client for the first few minutes after the device agent has been configured.

Verify the enrollment and inventory collection job

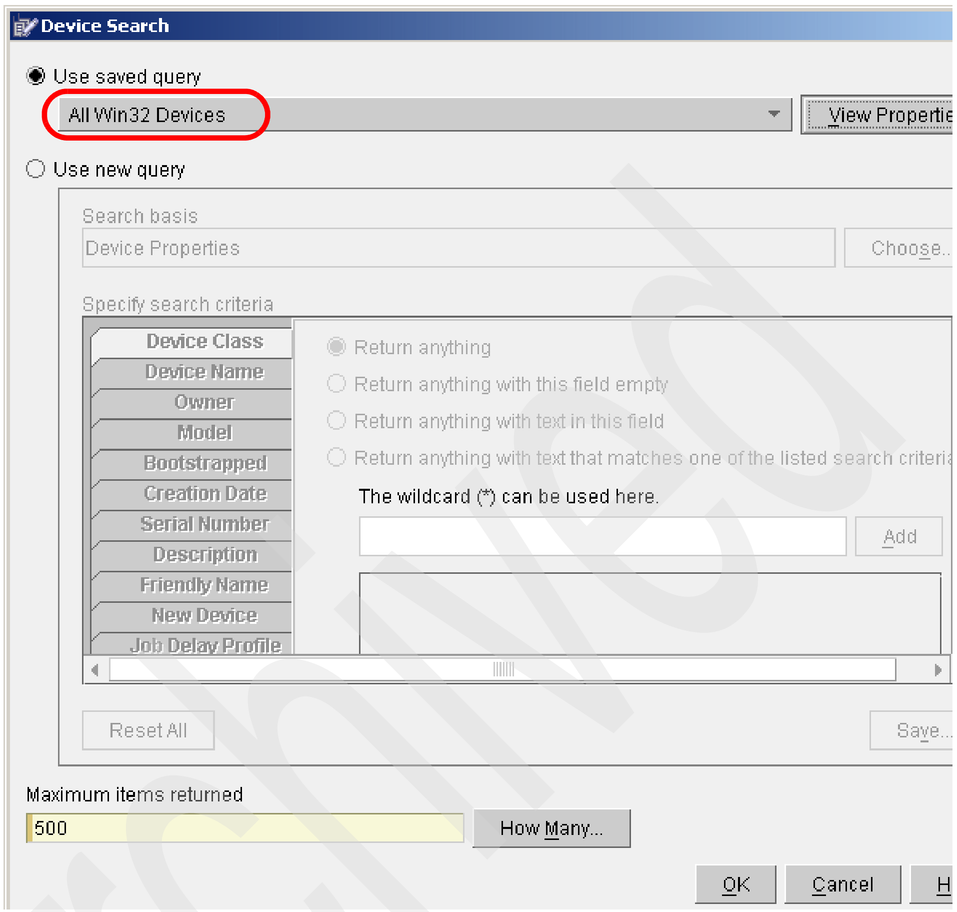
To verify that the WED client has been enrolled with the Device Manager server and the inventory collection job ran successfully, do the following:

1. Ensure the Device Manager server is started.
2. Launch the **Device Manager Console** and log in (see “Run and verify the Device Manager Console” on page 384).
3. Click **Devices** from the Device Manager Console and then click **OK**.
4. When the Device Search dialog appears, select the **Use saved query radio** button, select **All Win32 Devices** from the query drop-down list (as seen in Figure 14-7 on page 391), and then click **OK**.

Note: In our example we use WebSphere Everyplace Deployment Client for Windows and Linux as our client platform on a Windows XP-based system. The Enterprise Management Agent (device agent) included with this client platform extends the OSGi device agent with native Win32 or Linux extensions. This client platform is classified as a Win32 or Linux device class for this reason.

For this example, when using WebSphere Everyplace Deployment Client for Windows and Linux, the administrator should select the **Win32 Device** class, rather than the OSGi device class.

If the platform was Linux, the administrator would select Linux Device instead of Win32.



Device Search

● Use saved query

All Win32 Devices

○ Use new query

Search basis: Device Properties

Specify search criteria

● Return anything

○ Return anything with this field empty

○ Return anything with text in this field

○ Return anything with text that matches one of the listed search criteri

The wildcard (*) can be used here.

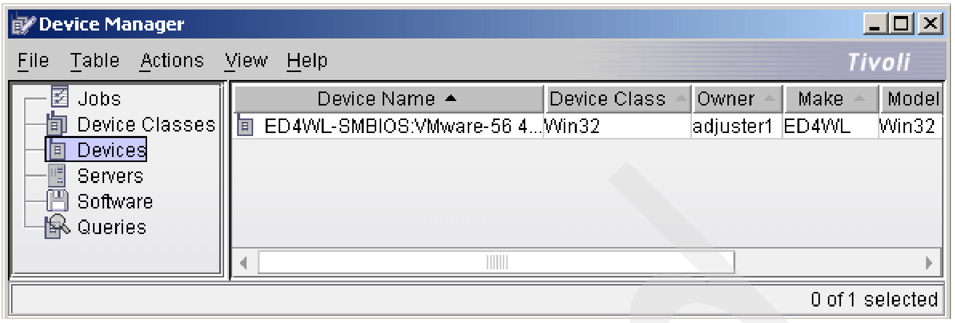
Maximum items returned: 500

Figure 14-7 Device Search - All Win32 Devices

You should see should see something like Figure 14-8 on page 392 displaying the device to verify the enrollment worked properly.

Note: In our example, we only have one device enrolled. If you have many devices, you can click the **View Properties** button to view how the search criteria is defined (SQL) for all Win32 devices.

The administrator could then potentially refine *Search criteria* (sql) to be used within the *New Query* search option.

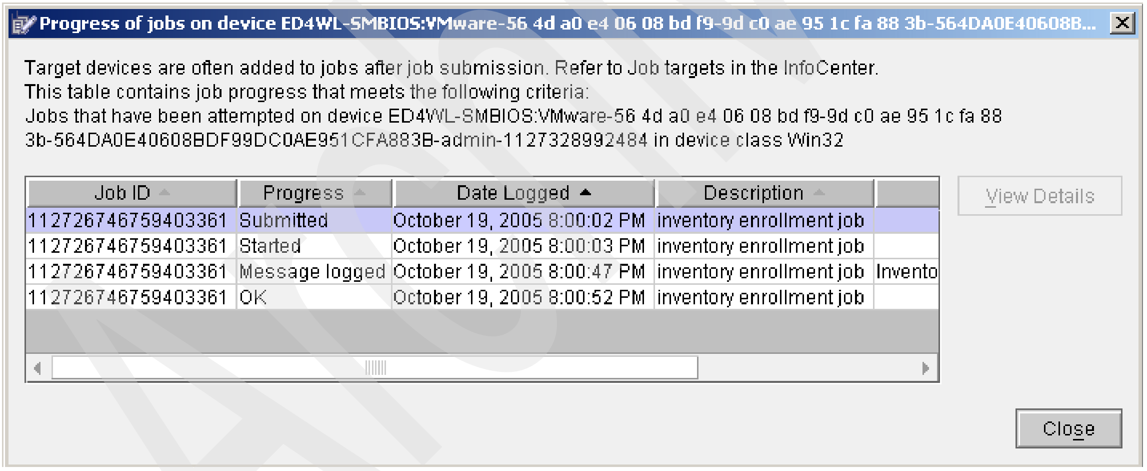


Device Name	Device Class	Owner	Make	Model
ED4WL-SMBIOS:VMware-56 4...	Win32	adjuster1	ED4WL	Win32

Figure 14-8 Device Manager - Search result of all Win32 devices

5. Right-click the desired device name found in the Win32 search (see Figure 14-8) and select **View Job Progress**.
6. When the Job Progress Search dialog appears, select **Return anything** and click **OK**.

   You should see something like Figure 14-9 showing the inventory job ran successfully.


Progress of jobs on device ED4WL-SMBIOS:VMware-56 4d a0 e4 06 08 bd f9-9d c0 ae 95 1c fa 88 3b-564DA0E40608B...
View Details
Close

Target devices are often added to jobs after job submission. Refer to Job targets in the InfoCenter.
This table contains job progress that meets the following criteria:
Jobs that have been attempted on device ED4WL-SMBIOS:VMware-56 4d a0 e4 06 08 bd f9-9d c0 ae 95 1c fa 88 3b-564DA0E40608BDF99DC0AE951CFA883B-admin-1127328992484 in device class Win32

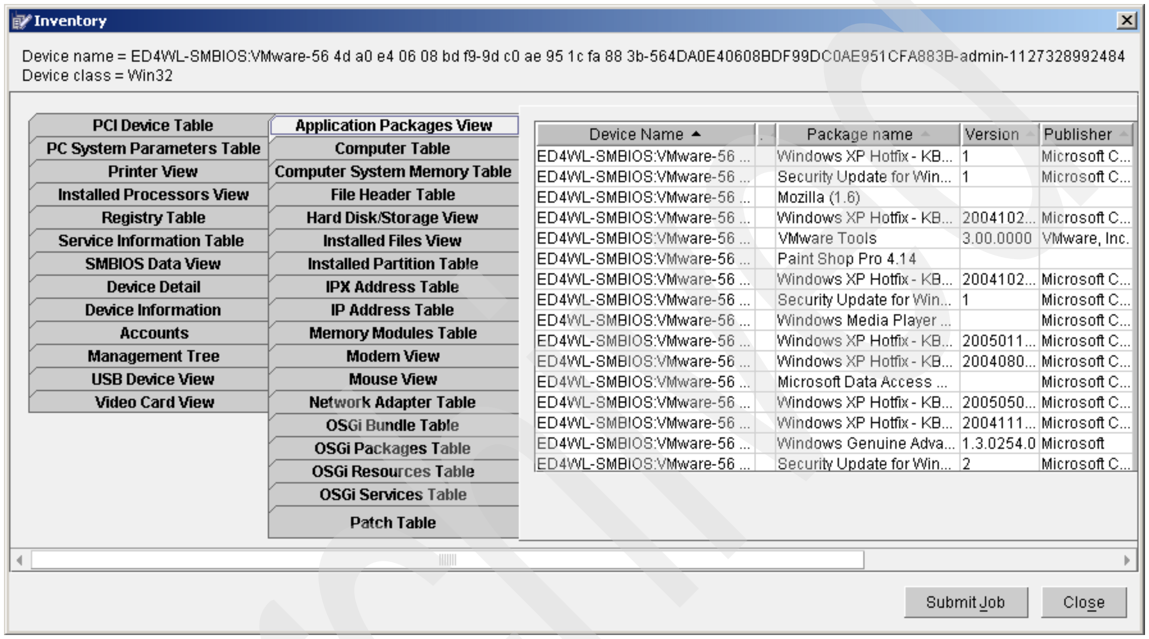
Job ID	Progress	Date Logged	Description	
112726746759403361	Submitted	October 19, 2005 8:00:02 PM	inventory enrollment job	
112726746759403361	Started	October 19, 2005 8:00:03 PM	inventory enrollment job	
112726746759403361	Message logged	October 19, 2005 8:00:47 PM	inventory enrollment job	Invento
112726746759403361	OK	October 19, 2005 8:00:52 PM	inventory enrollment job	

Figure 14-9 Progress of jobs - Inventory collection job

Troubleshooting tip: There are several different messages possible regarding job execution status. For example, if a inventory job could not be completed, you would see a `Failed - will retry` message. When the WED Client was restarted, the device agent attempts the job execution again. The Device Manager Console progress information can be very useful feedback to the administrator regarding job execution status.

7. Right-click the desired device name found in the Win32 search (see Figure 14-8 on page 392) and select **View Inventory**. The WED client device inventory details should be displayed as in Figure 14-10.

   When selecting the **Applications Packages View** tab, notice that native Windows applications are listed. This is possible since the Enterprise Management Agent extends the OSGi device agent with native Win32 extensions.



Device name = ED4WL-SMBIOS:VMware-56 4d a0 e4 06 08 bd f9-9d c0 ae 95 1c fa 88 3b-564DA0E40608BDF99DC0AE951CFA883B-admin-1127328992484
Device class = Win32

Device Name	.	Package name	Version	Publisher
ED4WL-SMBIOS:VMware-56 ...		Windows XP Hotfix - KB...	1	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Security Update for Win...	1	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Mozilla (1.6)		
ED4WL-SMBIOS:VMware-56 ...		Windows XP Hotfix - KB...	2004102...	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		VMware Tools	3.00.0000	VMware, Inc.
ED4WL-SMBIOS:VMware-56 ...		Paint Shop Pro 4.14		
ED4WL-SMBIOS:VMware-56 ...		Windows XP Hotfix - KB...	2004102...	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Security Update for Win...	1	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Windows Media Player ...		Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Windows XP Hotfix - KB...	2005011...	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Windows XP Hotfix - KB...	2004080...	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Microsoft Data Access ...		Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Windows XP Hotfix - KB...	2005050...	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Windows XP Hotfix - KB...	2004111...	Microsoft C...
ED4WL-SMBIOS:VMware-56 ...		Windows Genuine Adva...	1.3.0254.0	Microsoft
ED4WL-SMBIOS:VMware-56 ...		Security Update for Win...	2	Microsoft C...

Figure 14-10 Device Manager Console - Inventory

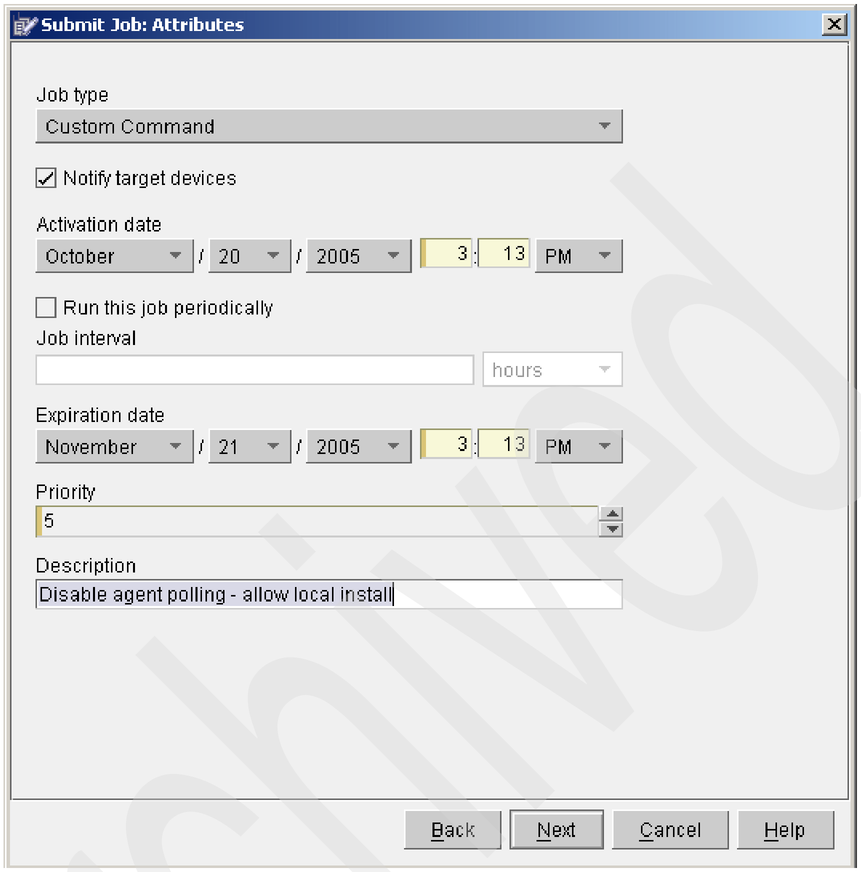
14.2.5 Create a custom command job (enable local install)

In “Configure the Enterprise Management Agent” on page 387, we explained that, once enabled, the user could not manually install applications on the local device, and this function would need to be managed from the server.

Imagine the scenario in which the user may want to manually install additional applications to her WebSphere Everyplace Deployment Client for Windows and Linux environment. It is possible for the administrator to disable the polling of the agent through a device configuration job, thereby giving the user the ability to perform her own installations and updates.

To create a custom command job in the Device Manager Console to disable polling, do the following:

1. Ensure the Device Manager server is started.
2. Launch the **Device Manager Console** and log in (see “Run and verify the Device Manager Console” on page 384).
3. Select **Devices** from the Device Manager Console.
4. When the Device Search dialog appears, select **All Win32 Devices** from the query drop-down list, and then click **OK**.
5. Right-click the desired device name, and select **Submit Job**.
6. When the Submit Job: Target Devices dialog appears, click **Next**.
7. When the Submit Job: Attributes dialog appears, do the following:
    a. Select **Custom Command** from the Job type drop-down list.
    b. Set the activation date desired.
    c. Enter `Disable agent polling - allow local install` in the Description field.
    d. Accept the defaults for the remaining fields, as seen in Figure 14-11 on page 395, and then click **Next**.


Submit Job: Attributes

Job type
Custom Command

[✓] Notify target devices

Activation date
October / 20 / 2005 3 : 13 PM

[ ] Run this job periodically
Job interval
hours

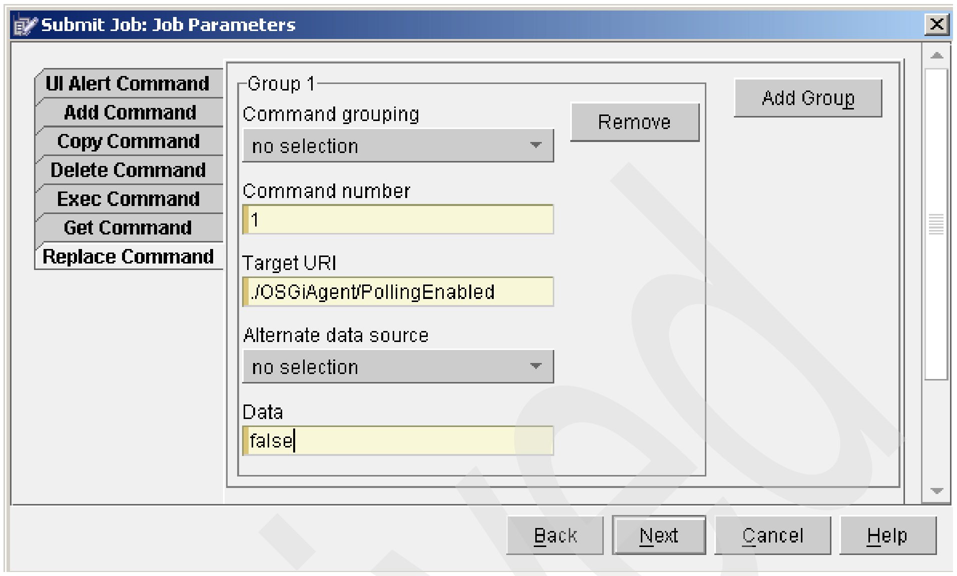
Expiration date
November / 21 / 2005 3 : 13 PM

Priority
5

Description
Disable agent polling - allow local install

Figure 14-11 Custom command job type example

8. When the Submit Job: Job Parameters dialog appears, do the following:
   a. Click the **Replace Command** tab, and then click **Add Group**.
   b. Enter `1` in the Command Number field.
   c. Enter `./OSGiAgent/PollingEnabled` in the Target URI field.
   d. Enter `false` in the Data field.
   e. Your dialog should look like Figure 14-12 on page 396. Click **Next**.
9. When the Submit Job: Summary dialog appears, click **OK**.



Group 1
Command grouping: no selection
Command number: 1
Target URI: ./OSGiAgent/PollingEnabled
Alternate data source: no selection
Data: false

Figure 14-12 Custom command window

10. You should see the message `Job created successfully`. Click **Close**.
11. Verify the WebSphere Everyplace Deployment Client for Windows and Linux Application Install and Management dialogs are enabled to allow for installing software and verify the job ran successfully.
    a. Start the WebSphere Everyplace Deployment Client for Windows and Linux on the client node.
    b. The Install and Application Management options should be selectable:
       - Select **Application** →**Install**.
       - Select **Application** → **Application Management**.
12. In addition, the "Enable Enterprise Management Agent" check box should no longer be checked. Verify by selecting **File** → **Preferences**.
13. Now that we have verified the use of the custom command job, ensure that you check the **Enable Enterprise Management Agent** check box, which is required for the remainder of this example.

14.2.6 Software distribution

The ITSO Insurance company is now ready to deploy the Mobile Adjuster application to the Adjuster devices (Windows XP laptops with WED Client) enrolled with the Device Manager server.

To begin with, the server administrator needs to obtain the Update Site zip file. Once this file is obtained, the administrator is going to create a native software bundle to deploy the software to the client. The Device Manager server includes a command-line utility, NativeAppBundle, to create this software bundle. This utility has the ability to wrap both native applications and WED client applications as an OSGi bundle. These can then be installed on the host device or into the client framework by the Enterprise Management Agent.

The following sections describe how to distribute software from the server to the client:

- Export the client-side application to an Update Site zip file.
- Create the OSGi NativeApp bundle.
- Create a DMS software distribution job.
- Client accepts MobileAdjuster software distribution job.

Export the client-side application to an Update Site zip file

The client-side application is packaged in an Eclipse bundle, known as an Update Site zip file. For the example, you have two options:

- Use the ITSO provided Update Site zip found in the c:\6775code\deploy\client\MobileAdjuster-UpdateSite.zip directory.
- Or export the client-side application to an Update Site zip.

Note: For details on exporting the client-side application to an Update Site zip, refer to 11.4.2, “Create the client-side application Update Site zip file” on page 320.

Create the OSGi NativeApp bundle

The Device Manager server requires that the Update Site zip file be packaged as an OSGi NativeApp bundle.

To create the OSGi NativeApp bundle, do the following:

1. Create the C:\MobileAdjusterClient\UpdateSite directory on the WED Server node.
2. Copy the MobileAdjuster-UpdateSite.zip to the C:\temp directory.

   Refer to “Export the client-side application to an Update Site zip file” on page 397 for instructions for obtaining the MobileAdjuster-UpdateSite.zip.
3. Unzip the c:\temp\MobileAdjuster-UpdateSite.zip in the C:\MobileAdjusterClient\UpdateSite directory.

Note: The NativeAppBundle command run in a subsequent step requires that the files in the Update Site are unzipped to the file system.

4. On the Access Server node (WED Server), open a Windows command window and navigate to the Device Manager home ..\bin directory; for example:

   ```
   C:\IBM\WebSphere\DMS18\bin
   ```

5. Enter the NativeAppBundle command (as seen in Example 14-1) to create the necessary OSGi NativeApp bundle for distribution through the Device Manager server.

Note: The following command (see Example 14-1) should be on one line in the command prompt. The command takes several minutes to execute.

We suggest that you enter this command in a batch file for future use; for example, we created C:\6775code\deploy\client\nab.bat containing the command listed in Example 14-1.

Example 14-1 ITSO sample NativeAppBundle command

```
NativeAppBundle -BundleName=MobileAdjuster
-InputDirectory=C:\MobileAdjusterClient\UpdateSite
-BuildDirectory=C:\MobileAdjusterClient -InstallDirectory=C:\\MobileAdjusterApp
-BundleVersion=6.0 -Eclipse=default
```

Where:

– *BundleName* is name of the bundle that will be visable within the Device Manager Console.

– *InputDirectory* is the location on the server where the MobileAdjuster-UpdateSite.zip has been copied.

– *BuildDirectory* is the location on the server where the new bundle will be placed.

– *InstallDirectory* is the path on the client computer to install the application.

Note: Do not forget the double backslashes.

– *BundleVersion* identifies the application version that will be displayed in Device Manager Console.

– *Eclipse=default* identifies this bundle as an Eclipse application.

6. Verify that the OSGi NativeApp bundle for the MobileAdjuster application was successfully created:

   a. You should see the following output message from running the NativeAppBundle command:

   ```
   SMF bundle: C:\MobileAdjusterClient\\MobileAdjuster+6_0.jar created
   successfully.
   ```

   b. Navigate to the C:\MobileAdjusterClient directory and verify the MobileAdjuster+6_0.jar OSGi NativeApp bundle file exists.

Create a DMS software distribution job

This section describes how to create the DMS software distribution job for the OSGi NativeApp bundle within the Device Manager Console.

1. Create the sw directory in the <IHS_home>\htdocs\<locale>\dms directory, for example, C:\IBM\HTTPServer\htdocs\en_US\dms\sw.
2. Copy the newly created MobileAdjuster+6_0.jar OSGi NativeApp bundle to the <IHS_home>\htdocs\<locale>\dms\sw directory so that the Device Manager server can access the OSGi NativeApp bundle:

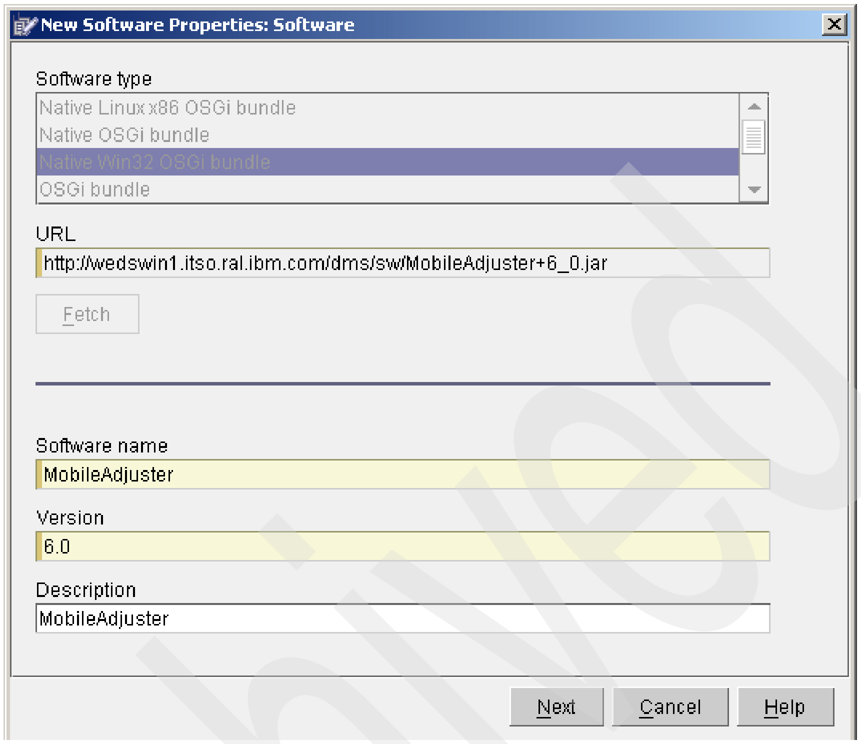
   ```
   cd \MobileAdjusterClient
   copy MobileAdjuster-UpdateSite.jar C:\IBM\HTTPServer\htdocs\en_US\dms\sw
   ```

3. Launch the **Device Manager Console** and log in (for details see "Run and verify the Device Manager Console" on page 384).
4. Click **Software** and then click **OK**.
5. Right-click **Software** and select **New Software**.
6. When the New Software Properties: Software window appears, do the following:

   a. Select **Native Win32 OSGi Bundle** from the Software Type list.

   b. Enter `http://wedwin1.itso.ral.ibm.com/dms/sw/MobileAdjuster+6_0.jar` in the URL field.

   c. Click **Fetch** to verify the URL is working correctly. You should then see a dialog like Figure 14-13 on page 400 (Software name, Version, Description).



New Software Properties: Software

Software type

- Native Linux x86 OSGi bundle
- Native OSGi bundle
- Native Win32 OSGi bundle
- OSGi bundle

URL: http://wedswin1.itso.ral.ibm.com/dms/sw/MobileAdjuster+6_0.jar

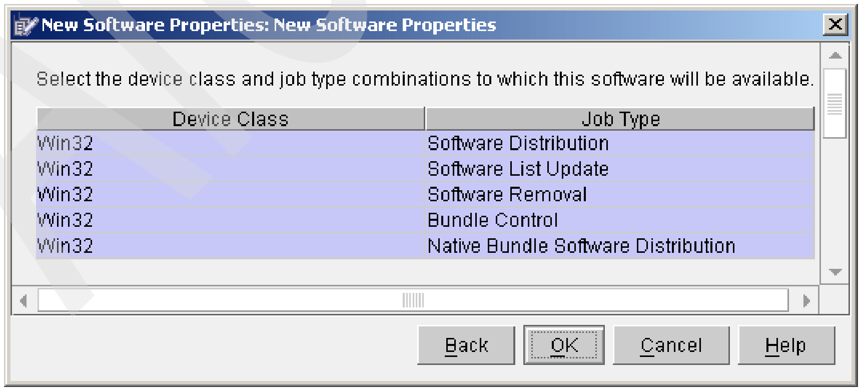
Software name: MobileAdjuster

Version: 6.0

Description: MobileAdjuster

Figure 14-13 Software Distribution window

7. Click **Next**.
8. When the Select device class dialog appears, accept the default job types for the MobileAdjuster application, as seen in Figure 14-14, and click **OK**.



New Software Properties: New Software Properties

Select the device class and job type combinations to which this software will be available.

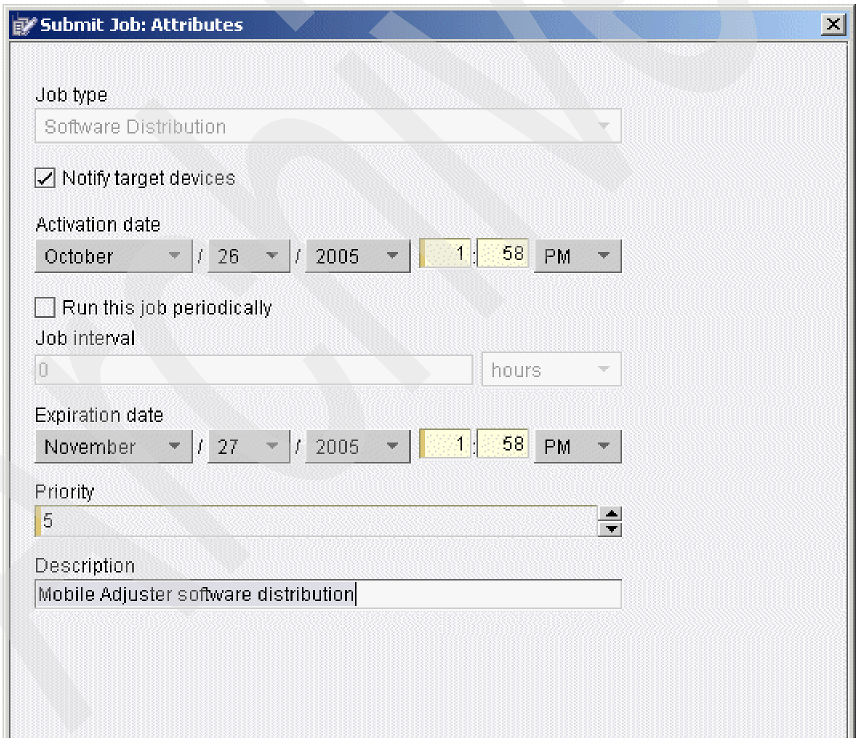
Device Class	Job Type
Win32	Software Distribution
Win32	Software List Update
Win32	Software Removal
Win32	Bundle Control
Win32	Native Bundle Software Distribution

Figure 14-14 Device classes - Default job types

9. Right-click the **MobileAdjuster** software entry and select **Submit Job**.
10. When the Submit Job: Target Devices dialog appears, do the following:
    a. Select **Win32** in the Device class drop-down list.
    b. Select **DB2eAdjusters** in the Owner group drop-down list.

    > **Note:** Optionally, you can confirm the devices this software will be installed on by clicking **View Devices** and then **OK** in the pop-up window. For now there should be one device in the resulting list, belonging to adjuster1.

    c. Click **Next**.
11. When the Submit Job: Attributes dialog appears, we entered `MobileAdjuster software distribution job` in the Description field, and accepted the default settings for the remaining fields (see Figure 14-15). Click **Next**.



Job type: Software Distribution
☑ Notify target devices
Activation date: October / 26 / 2005 1 : 58 PM
☐ Run this job periodically
Job interval: 0 hours
Expiration date: November / 27 / 2005 1 : 58 PM
Priority: 5
Description: Mobile Adjuster software distribution

Figure 14-15 Submit Job attributes dialog

12. Click **Add Group**. We accepted the default settings and clicked **Next**.

13. When the Submit Job: Summary dialog appears, click **OK**.

14. You should see the message `Job create successfully`. Click **Close**.

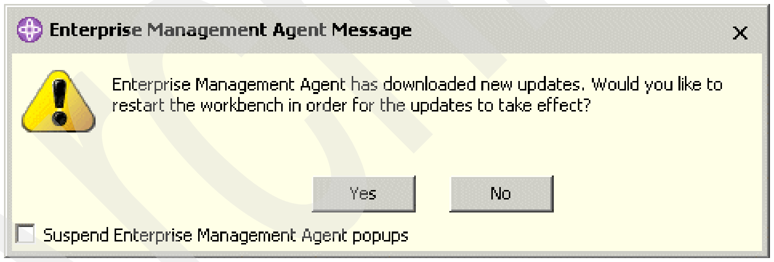
Client accepts MobileAdjuster software distribution job

The Mobile Adjuster software is now cataloged on the Device Manager server and is ready to be distributed as a software distribution job to all users in the DB2eAdjusters group on the WebSphere Everyplace Deployment Server. Since we created the job to be distributed to all existing and enrolled devices, any new users who are added to the DB2eAdjusters user group and who have devices with the WED Client installed and Enterprise Management Agent configured will receive the software as the client enrolls with the Device Manager server.

Since the software job is created and the user's client is already enrolled with the Device Manager server, it is just a matter of waiting for the Enterprise Management Agent on the user's client to poll the Device Manager server to receive the job. Once the software job has been received on the client a pop-window with the WED client appears.

To install the MobileAdjuster client-side application on the WED Client via the software distribution job initiated from the server, do the following:

1. You should see a flashing icon in the system tray of Windows XP. If you double-click the icon you will see the Enterprise Management Agent Message displayed in Figure 14-16. Click **Yes**.


Enterprise Management Agent Message
Yes
No
Suspend Enterprise Management Agent popups

Enterprise Management Agent has downloaded new updates. Would you like to restart the workbench in order for the updates to take effect?

Figure 14-16 Enterprise Management Agent message - New updates

Tip: When restarting the WED Client, the Enterprise Management Agent immediately polls the Device Manager server for available jobs. For testing purposes, we suggest restarting the WED client to expedite the process.

2. Once WebSphere Everyplace Deployment Client for Windows and Linux has restarted, click **Open** in the left frame of the workbench to see the installed applications.

3. Configure the MobileAdjuster.

Refer to “Configure the MobileAdjuster application” on page 372 for configuration details.

4. Click **MobileAdjuster** in the list and verify the application is now installed and ready to use.

   For details refer to Chapter 15, “Application walkthrough” on page 407.

Note: For the new software to show up in the DMS inventory, a new inventory job will need to be run.

Troubleshooting software distribution jobs

When testing or troubleshooting, you may want to run the software distribution job again. You will need to do the following if you desire to use the same Software name (contained in the NativeApp Bundle .jar file):

1. Create a Software Removal job and verify the job is run on the client.
2. Cancel the Software Distribution job in DM Console.
3. Delete the Software in DM Console.
4. Recatalog the Software in DM Console.
5. Create a new Software Distribution job in DM Console.

14.2.7 Device configuration

The ITSO Insurance company has decided to enable SSL for the IBM HTTP Server on the WED Server. Now the administrator must create a device configuration job for all clients that will change the server address of the Enterprise Management Agent for each client to use HTTPS.

Note: For testing purposes, you can run this device configuration job without the IBM HTTP Server being enabled for SSL. When done running the device configuration job, the Enterprise Device Management agent should have the HTTPS protocol selected. Simply select HTTP and click **OK** to restore to the original settings.

1. Launch the Device Manager Console.
2. Click **Device Classes**.
3. Right-click **Win32** and select **Submit Job**.
4. Select the **Currently Enrolled** radio button to only configure those devices currently enrolled with the device manager server and click **Next**.
5. When the Submit Job Attributes dialog appears, do the following:
   a. Select **Device Configuration** for Job type.

   b. Enter `Change Enterprise Management Agent to SSL` in the Description field.

   c. Click **Next**.

6. Click **Add Group**.

   a. Select **Modify** from the Action drop-down list.

   b. Enter `SampleAccount` in the Name field.

   > **Note:** The name will be the same for all WebSphere Everyplace Deployment Client for Windows and Linux clients enrolled with the Device Manager server.

   c. Enter the following in the Address field:

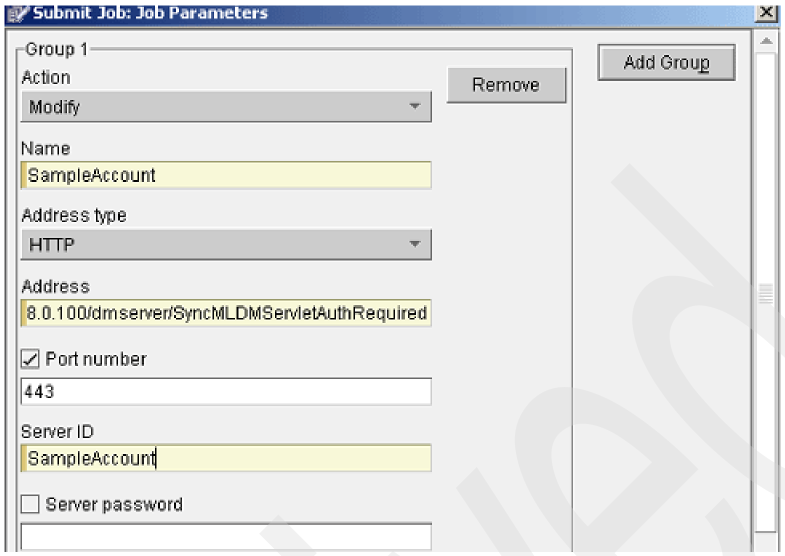
   `https://wedswin1.itso.ral.ibm.com/dmserver/SyncMLDMServletAuthRequired`

   d. Enter `443` in the Port number field.

   e. Enter `SampleAccount` in the Server ID field.

   > **Note:** The name will be the same for all WebSphere Everyplace Deployment Client for Windows and Linux clients enrolled with the Device Manager server.

   f. When done, the Add Group page should look like Figure 14-17 on page 405. Click **Next** to continue.



Group 1

Action: Modify

Name: SampleAccount

Address type: HTTP

Address: 8.0.100/dmserver/SyncMLDMServletAuthRequired

[✓] Port number: 443

Server ID: SampleAccount

[ ] Server password

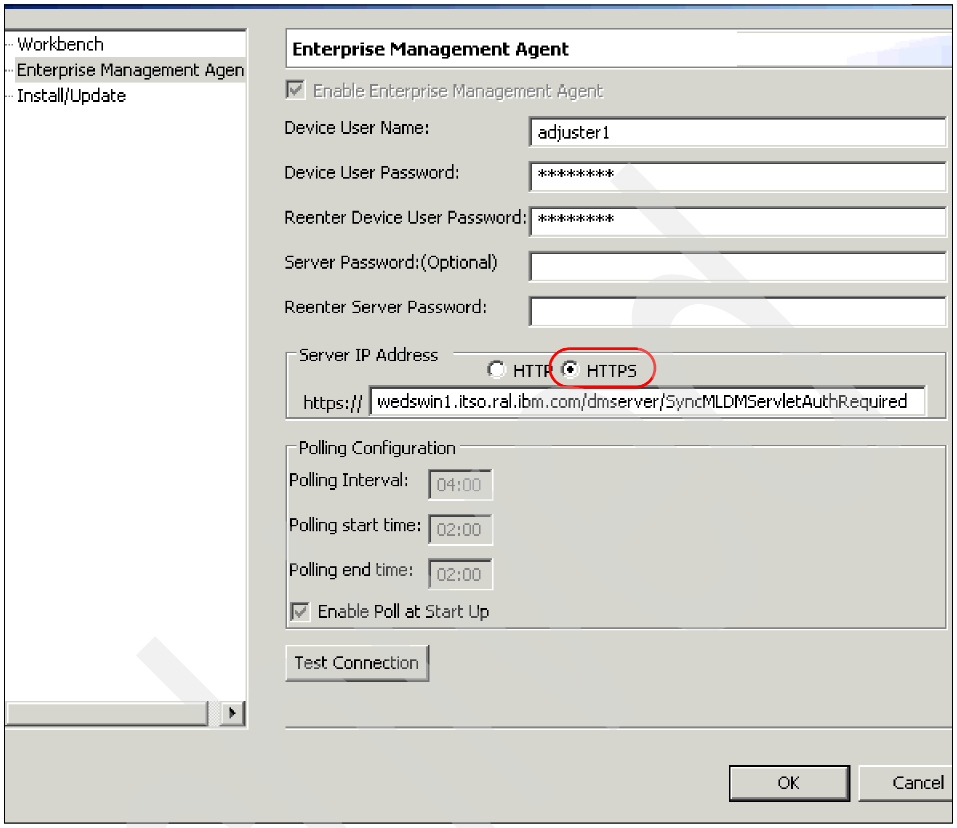
Figure 14-17 Device Configuration window

7. Click **OK**.
8. Verify the device configuration job ran properly.

   Since the Enterprise Management Agent on the client is set to poll the server automatically, the next time this action is performed the changes will appear.

   Note: When restarting the WED Client, the Enterprise Management Agent will immediately poll the Device Manager server for available jobs. For testing purposes, we suggest restarting the WED client to expedite the process.

   a. Launch IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux.
   b. Select **File** → **Preferences** from the WebSphere Everyplace Deployment menu.
   c. Select **Enterprise Management Agent**.
   d. When the Enterprise Management Agent dialog appears, ensure the HTTPS radio button is selected, as seen in Figure 14-18 on page 406.


Workbench
Enterprise Management Agen
Install/Update

Enterprise Management Agent

[✓] Enable Enterprise Management Agent

Device User Name:	adjuster1
Device User Password:	*********
Reenter Device User Password:	*********
Server Password:(Optional)	
Reenter Server Password:	

Server IP Address
HTTP [✓ HTTPS]
https:// wedswin1.itso.ral.ibm.com/dmserver/SyncMLDMServletAuthRequired

Polling Configuration

Polling Interval:	04:00
Polling start time:	02:00
Polling end time:	02:00

[✓] Enable Poll at Start Up

Figure 14-18 Enterprise Management settings after Device Configuration job

Note: If you are running the device configuration for testing purposes, simply select HTTP and click **OK** to reset.

15

Application walkthrough

This chapter provides an application walkthrough of the ITSO Insurance system for a an end-to-end customer claims processing scenario. We exercise use cases in both the Mobile Agent online Web application accessible via a Web browser, and the disconnectable Mobile Adjuster application running in WebSphere Everyplace Deployment Client for Windows and Linux on the adjuster laptop.

The objective of the chapter is to validate the use cases defined in 8.3, “Use cases” on page 204, for the ITSO working example, and highlight the underlying capabilities (local Web UI, DB2e data sync, MQe transactional messaging, Web services on client) that define the Distributed Rich Client application pattern.

The chapter is organized into the following sections:

- Prerequisites for application walkthrough
- UC_1: Create new claim
- UC_2: Log on to disconnectable client application
- UC_3: Retrieve new claims
- UC_4: Enter claim data
- UC_6: Upload image
- UC_7: View claims
- UC_8: View customer profile
- UC_9: Sync claims data
- UC_10: Refresh claim status

15.1 Prerequisites for application walkthrough

Before performing the application walkthrough, ensure that the following prerequisites have been met:

- Runtime environment is installed and configured.

  For details refer to Chapter 12, “Runtime environment installation” on page 325.
- ITSO MobileAdjuster application is deployed.

  For details refer to Chapter 13, “Application deployment” on page 345.
- The servers are started.

  For details refer to “Server startup” on page 408 (below).

15.1.1 Server startup

In order to run the ITSO MobileAdjuster application, the following servers must be started:

- IBM HTTP Server
- WebSphere Everyplace Deployment servers
  - coreServices
  - DMS_AppServer
  - IBMDB2eServer
- server1 application server (server where ITSO MobileAdjuster enterprise application is deployed)
- MQe Claim Server

Start IBM HTTP Server

Ensure the IBM HTTP Server Windows service is started.

Start WebSphere Everyplace Deployment servers

The WebSphere Everyplace Deployment servers are application servers that run within the WebSphere Application Server. They can be started as a group or individually.

To start all WebSphere Everyplace Deployment servers, click **Start** → **All Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **Start Servers**.

Note: After WebSphere Everyplace Deployment is installed, WebSphere security is enabled; thus, a valid administrator user ID and password will be required to stop the application servers.

To stop all WebSphere Everyplace Deployment servers, click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Everyplace Deployment** → **Stop Servers**.

You will be prompted to enter the WebSphere Everyplace Deployment administrator ID and password.

Or, to start the WebSphere Everyplace Deployment servers individually, do the following:

1. Open a command window and navigate to the following directory:

   ```
   C:\IBM\WebSphere\AppServer\profiles\default\bin
   ```

2. Enter the following command to check the status of the servers:

   ```
   serverstatus -all
   ```

   You should see a status list for each of the following application servers:

   ```
   coreServices
   DMS_AppServer
   IBMDB2eServer
   ```

3. Enter the following command to start the application servers:

   ```
   startserver coreServices
   ```

4. Repeat this process for each of the application servers listed.

Start server1

To start server1 do one of the following:

- Click **Start** → **Programs** → **IBM WebSphere** → **WebSphere Application Server** → **Start Servers**.
- To start server1 from the command line, enter the following:

  ```
  cd C:\IBM\WebSphere\AppServer\profiles\default\bin
  startserver server1
  ```

Note: To stop server1, enter the following:

```
stopserver server1 -username wedadmin -password password
```

Start the MQ Everyplace Claim Server

To run the MQ Everyplace Claim Server, do the following:

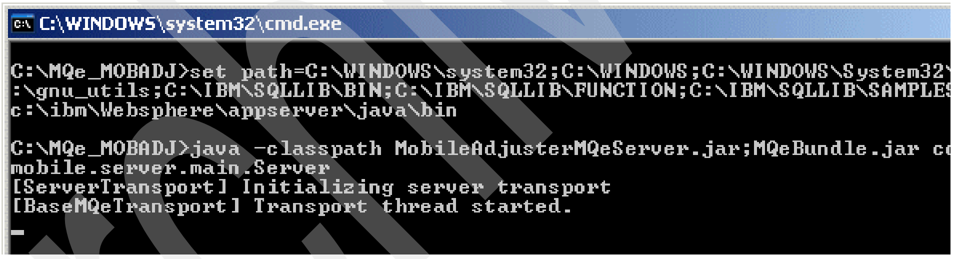
1. Open a command window and navigate to the C:\MQe_MOBADJ directory.
2. Set the Java environment. You can either add Java to the system path or define it manually at the command line as follows:

   ```
   set path=%path%;C:\IBM\WebSphere\AppServer\java\bin
   ```

   > **Note:** We recommend adding the above statement to the top of the runMQeServer.bat found in the C:\MQe_MOBADJ. Alternatively, add the Java environment to the system PATH environment variable.

3. Enter `runMQeServer.bat` to start the server. You should see something like Figure 15-1.

> **Note:** The first time you start the MobileAdjuster MQ Everyplace Server you will see an exception because the queue manager does not exist; thus, it will create the queue manager.



```
C:\WINDOWS\system32\cmd.exe

C:\MQe_MOBADJ>set path=C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32
:\gnu_utils;C:\IBM\SQLLIB\BIN;C:\IBM\SQLLIB\FUNCTION;C:\IBM\SQLLIB\SAMPLES
c:\ibm\Websphere\appserver\java\bin

C:\MQe_MOBADJ>java -classpath MobileAdjusterMQeServer.jar;MQeBundle.jar co
mobile.server.main.Server
[ServerTransport] Initializing server transport
[BaseMQeTransport] Transport thread started.
_
```

Figure 15-1 MQ Everyplace Claim Server

> **Note:** To stop the server, press Ctrl+C.

15.2 UC_1: Create new claim

There are two methods of creating a new claim. The primary method is for the agent to create the new claim in the Mobile Agent Web application. Alternatively, the adjuster can create the claim on-site with the customer in the disconnectable Mobile Adjuster application, and synchronize the claim data to the enterprise when connected to a network.

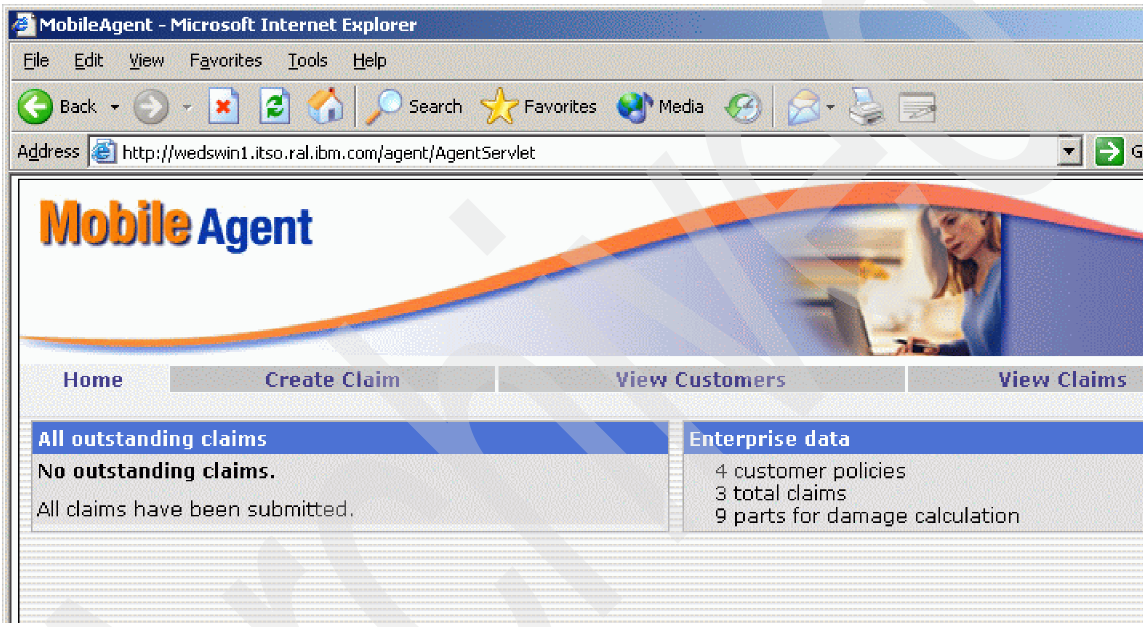
Create new claim (agent - Mobile Agent Web application)

To create a new claim in the role of an agent in the Mobile Agent Web application, do the following:

1. Enter the following URL in a Web browser to launch the Mobile Agent online Web application:

   `http://wedswin1.itso.ral.ibm.com/agent/AgentServlet`

2. When the Mobile Agent home page appears, click **Create Claim**, as seen in Figure 15-2.


MobileAgent - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://wedswin1.itso.ral.ibm.com/agent/AgentServlet
Mobile Agent
Home
Create Claim
View Customers
View Claims

All outstanding claims

No outstanding claims.

All claims have been submitted.

Enterprise data

4 customer policies
3 total claims
9 parts for damage calculation

Figure 15-2 Online application main screen

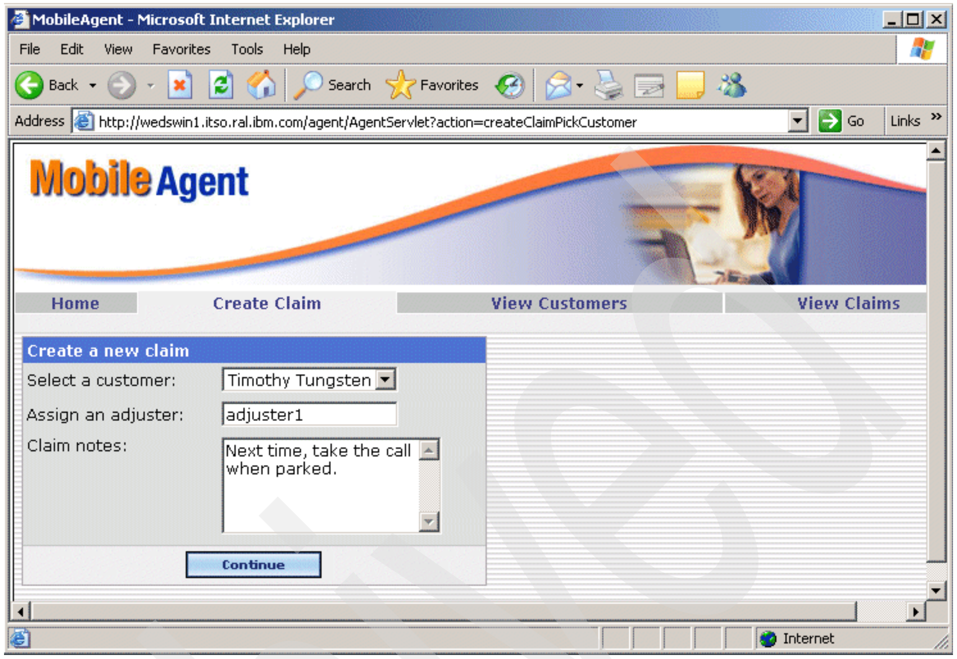
3. When the Create a new claim page appears, do the following, as seen in Figure 15-3 on page 412:

   a. Select the customer name (for example, Timothy Tungsten) from the Customer name drop-down list.

   b. Assign the adjuster to the claim by entering the adjuster name (for example, adjuster1) in the Adjuster name field.

   > **Note:** The Adjuster name field is case sensitive. In our example, we created a user ID named adjuster1.

   c. Click **Continue**.


MobileAgent - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://wedswin1.itso.ral.ibm.com/agent/AgentServlet?action=createClaimPickCustomer
Go Links
Mobile Agent
Home
Create Claim
View Customers
View Claims
Continue
Internet

Create a new claim

Select a customer: Timothy Tungsten

Assign an adjuster: adjuster1

Claim notes: Next time, take the call when parked.

Figure 15-3 Create new claim

Create new claim (adjuster - disconnectable Mobile Adjuster)

As an alternative, the adjuster can create a new claim in the disconnectable Mobile Adjuster application, and synchronize the claim data to the enterprise when connected to the network.

1. Start WebSphere Everyplace Deployment Client for Windows and Linux (client).
2. Click **Open** and select **Mobile Adjuster**.
3. When the Login dialog appears, enter the User Name and Password (for example, adjuster1) and then click **Login**.

> **Note:** The user name is case sensitive. The credentials are stored and used by DB2e to authenticate before synchronizing the claim data to the enterprise. If you do not enter valid credentials the sync will fail.

4. Click **Create Claim**.
5. When the Select customer page appears, select the customer from the list (for example, Timothy Tungsten) and then click **Select**.

Note: Since the adjuster is automatically assigned the claim since they opened the claim, the create claim page also includes Add claim information fields. For details refer to 15.5, “UC_4: Enter claim data” on page 417.

15.3 UC_2: Log on to disconnectable client application

For the adjuster to log on to the disconnectable Mobile Adjuster application, do the following:

1. Start WebSphere Everyplace Deployment Client for Windows and Linux (client).
2. Click **Open**, and select **Mobile Adjuster**, as seen in Figure 15-4.

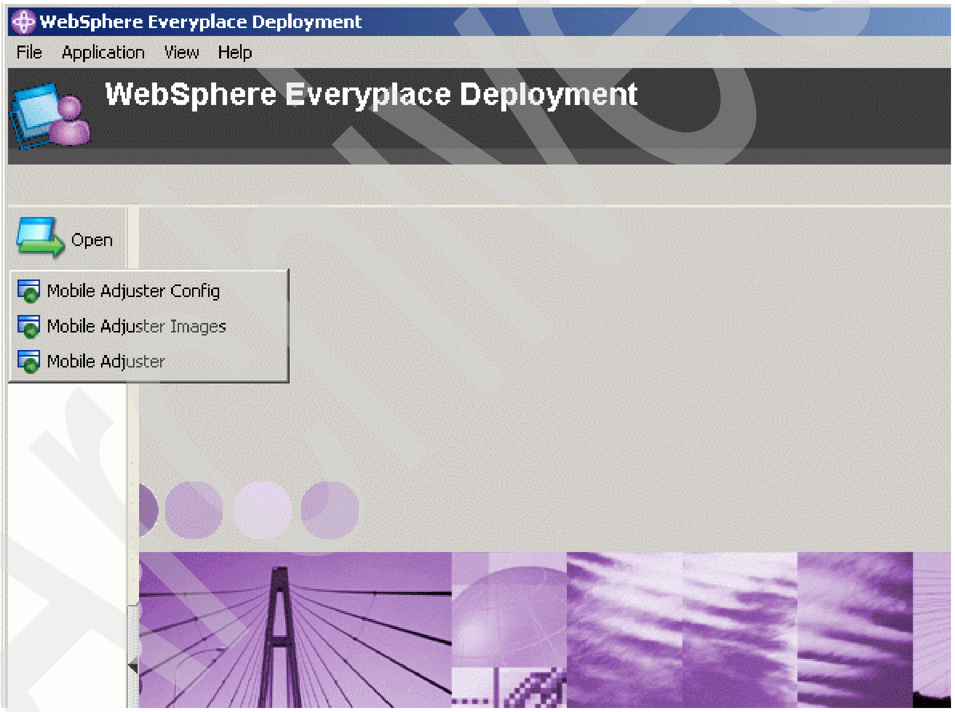


Figure 15-4 WebSphere Everyplace Deployment main screen

3. When the Login dialog appears, enter the User Name and Password (for example, adjuster1), as seen in Figure 15-5 on page 414, and then click **Login**.

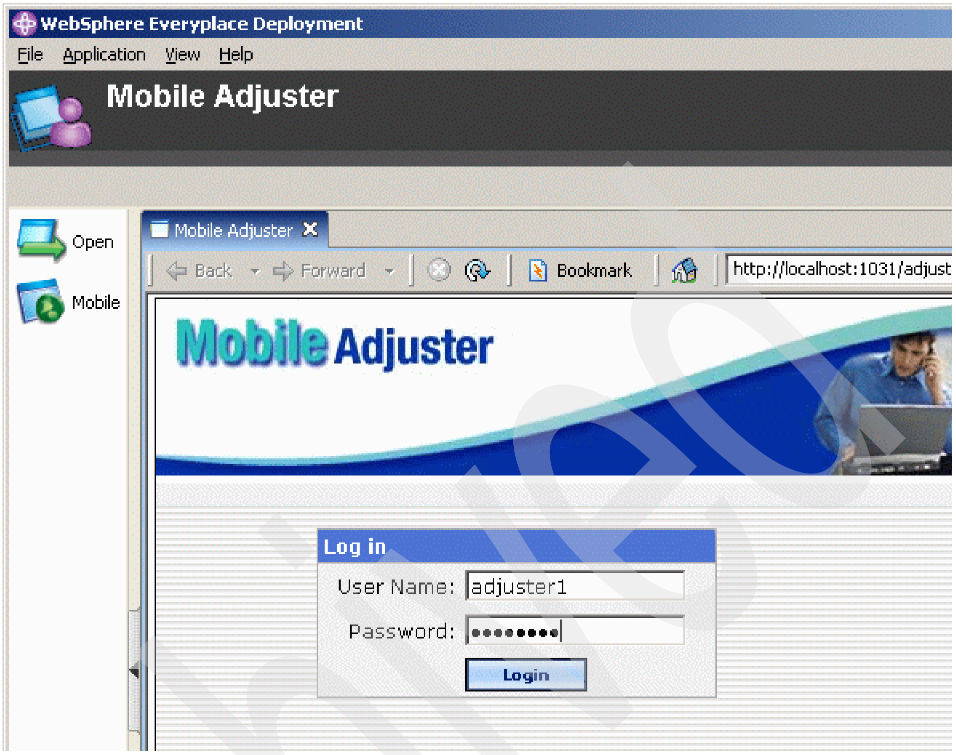
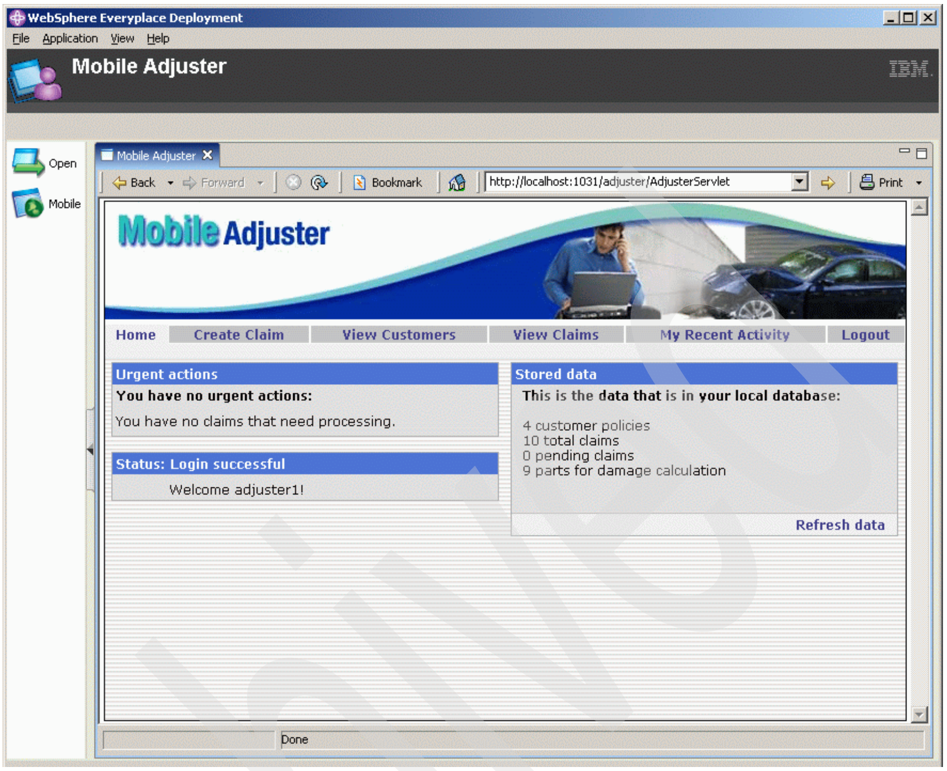


Figure 15-5 Enter username and password

> **Note:** The user name is case sensitive. The credentials are stored and used by DB2e to authenticate before synchronizing the claim data to the enterprise. If you do not enter valid credentials the synchronization will fail in a later step when clicking Refresh Data.

After performing the Login, you should see a page like Figure 15-6 on page 415.


WebSphere Everyplace Deployment
File Application View Help
Mobile Adjuster
IBM
Open
Mobile
Mobile Adjuster
Back Forward Bookmark Print
http://localhost:1031/adjuster/AdjusterServlet
Mobile Adjuster
Home
Create Claim
View Customers
View Claims
My Recent Activity
Logout

Urgent actions

You have no urgent actions:

You have no claims that need processing.

Status: Login successful

Welcome adjuster1!

Stored data

This is the data that is in your local database:

4 customer policies
10 total claims
0 pending claims
9 parts for damage calculation

Refresh data

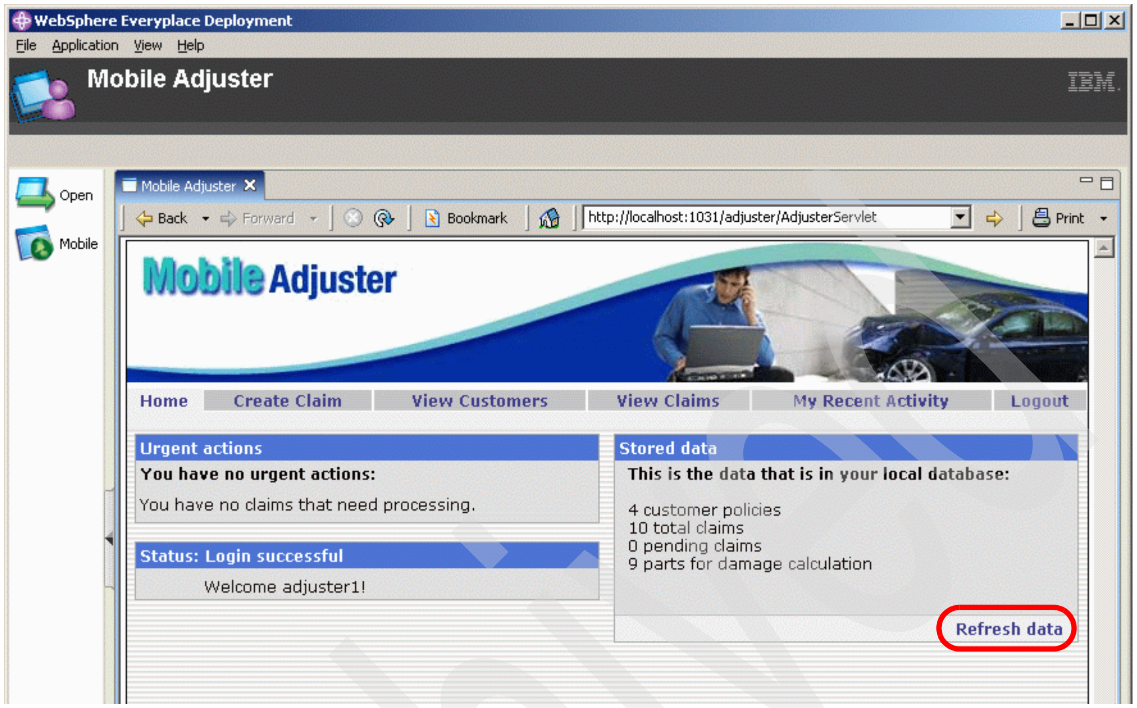
Done

Figure 15-6 Home page of disconnectable Mobile Adjuster application

15.4 UC_3: Retrieve new claims

After the adjuster has successfully logged on to the disconnectable Mobile Adjuster application, he can retrieve new claims assigned to him from the enterprise.

1. Click **Home** in the Mobile Adjuster application.



Urgent actions

You have no urgent actions:

You have no claims that need processing.

Status: Login successful

Welcome adjuster1!

Stored data

This is the data that is in your local database:

4 customer policies
10 total claims
0 pending claims
9 parts for damage calculation

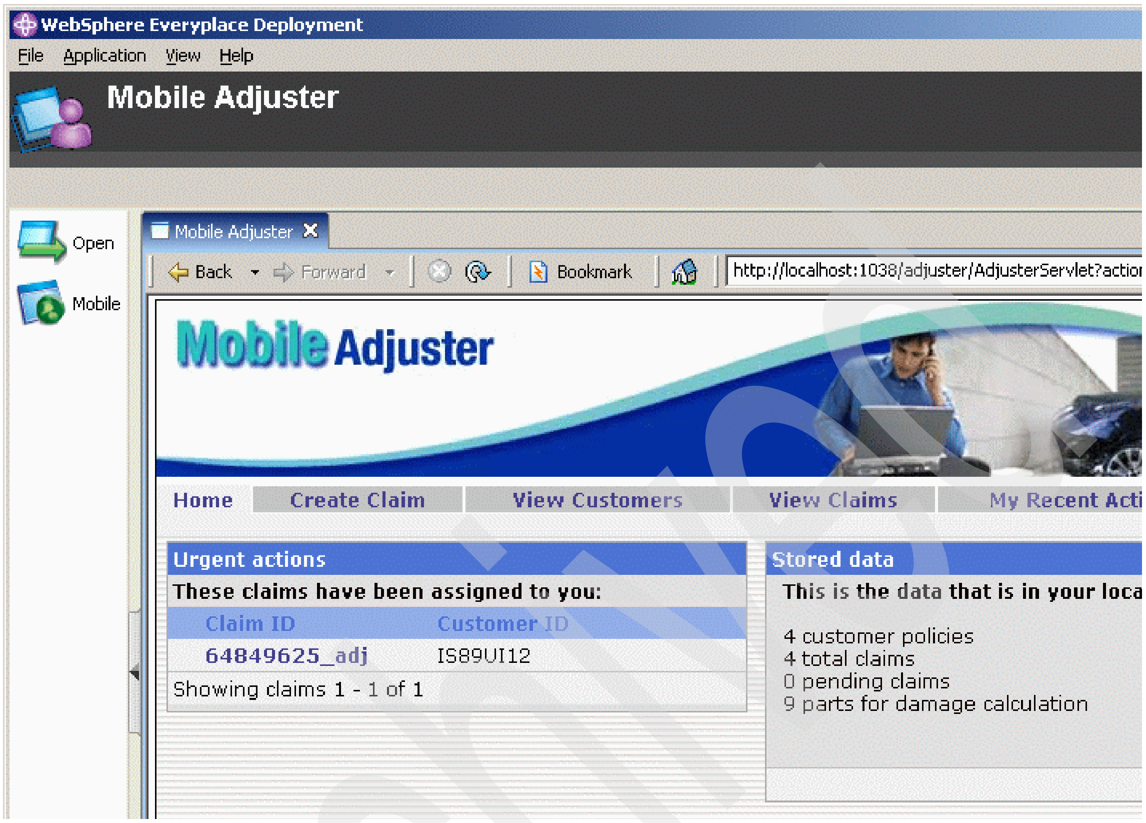
Figure 15-7 Home screen of Mobile Adjuster application

2. Click **Refresh data** in the bottom right corner to synchronize claims data (in this case to retrieve new claims from the enterprise).
3. You should see the message dialog `Data refreshed successfully`. Click **OK**.

   The new claim should appear under the claims assigned to you (adjuster1), as seen in Figure 15-8 on page 417.

Note: When configuring the DB2e subscription in MDAC, you can define the Timing value (measured in seconds) that subscriptions will be synchronized from the application database and the DB2e mirror database on the WED Server node.

By default, the subscription to synchronize the application database and DB2e mirror database is set to 3600 seconds. For our example scenario, we chose to set this to 120 seconds to facilitate the application walkthrough. For details refer to “Create a subscription set and assign to a group” on page 356.



Urgent actions

These claims have been assigned to you:

Claim ID	Customer ID
64849625_adj	IS89UI12

Showing claims 1 - 1 of 1

Stored data

This is the data that is in your loca

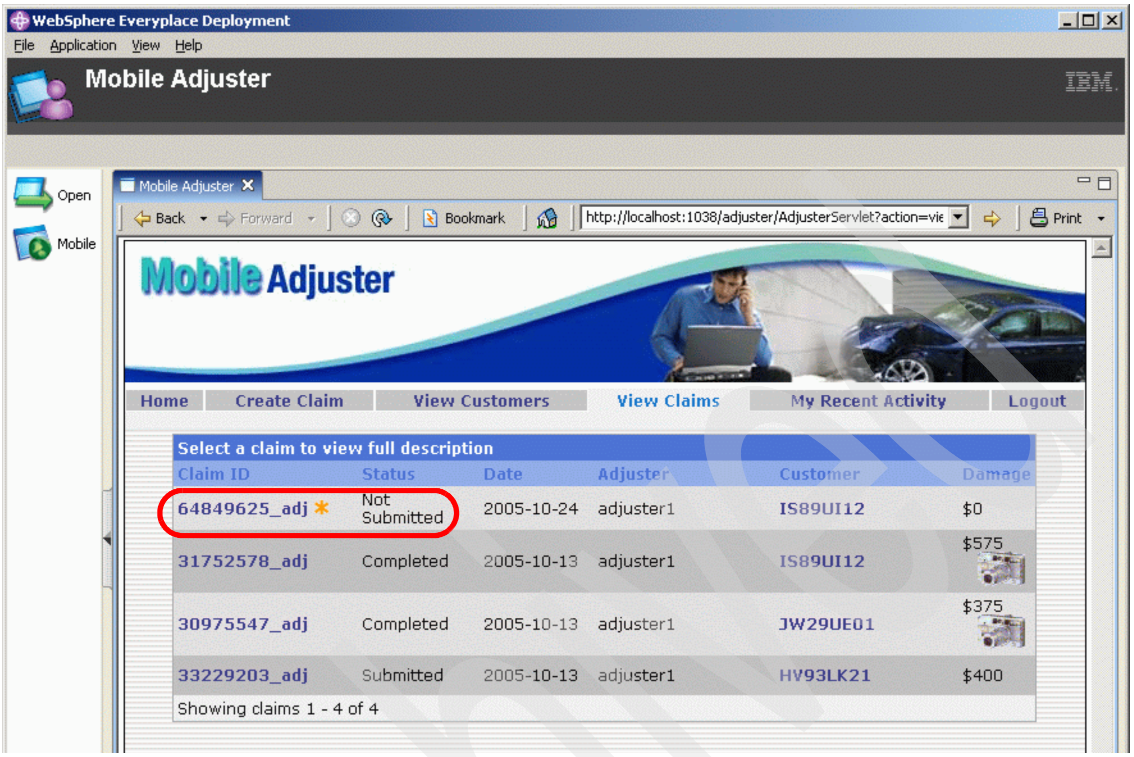
4 customer policies
4 total claims
0 pending claims
9 parts for damage calculation

Figure 15-8 New claim retrieved in the disconnectable Mobile Adjuster

15.5 UC_4: Enter claim data

To enter the claim data while on-site with the customer in the disconnectable Mobile Adjuster application in the role of the adjuster, do the following:

1. Click **View Claims** from the Mobile Adjuster home page.
2. Select the desired claim. The new claims are highlighted with an asterisk (*) and the Status is set to `Not Submitted`, as seen in Figure 15-9 on page 418.


WebSphere Everyplace Deployment
File Application View Help
Mobile Adjuster
Open
Mobile
Back Forward Bookmark Print
http://localhost:1038/adjuster/AdjusterServlet?action=vie
Mobile Adjuster
Home Create Claim View Customers View Claims My Recent Activity Logout

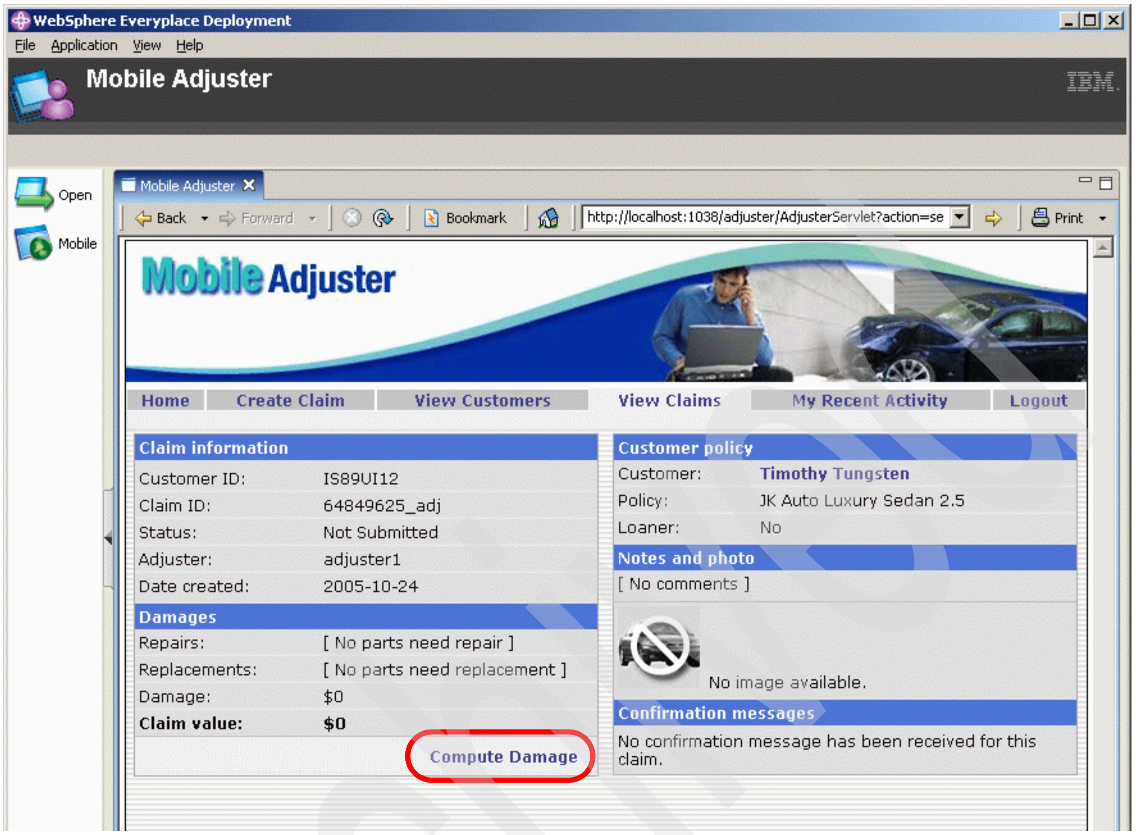
Select a claim to view full description

Claim ID	Status	Date	Adjuster	Customer	Damage
64849625_adj *	Not Submitted	2005-10-24	adjuster1	IS89UI12	$0
31752578_adj	Completed	2005-10-13	adjuster1	IS89UI12	$575
30975547_adj	Completed	2005-10-13	adjuster1	JW29UE01	$375
33229203_adj	Submitted	2005-10-13	adjuster1	HV93LK21	$400

Showing claims 1 - 4 of 4

Figure 15-9 View claims screen with new claim retrieved

3. When the Claim information details page is displayed, click **Compute Damage** (see Figure 15-10 on page 419) to open the claim form to enter the claim damage information.


WebSphere Everyplace Deployment
File Application View Help
Mobile Adjuster
IBM.
Open
Mobile
Mobile Adjuster
Back Forward Bookmark Print
http://localhost:1038/adjuster/AdjusterServlet?action=se
Mobile Adjuster
Home Create Claim View Customers View Claims My Recent Activity Logout
Compute Damage

Claim information

Customer ID:	IS89UI12
Claim ID:	64849625_adj
Status:	Not Submitted
Adjuster:	adjuster1
Date created:	2005-10-24

Damages

Repairs:	[No parts need repair]
Replacements:	[No parts need replacement]
Damage:	$0
Claim value:	**$0**

Customer policy

Customer:	Timothy Tungsten
Policy:	JK Auto Luxury Sedan 2.5
Loaner:	No

Notes and photo

[No comments]

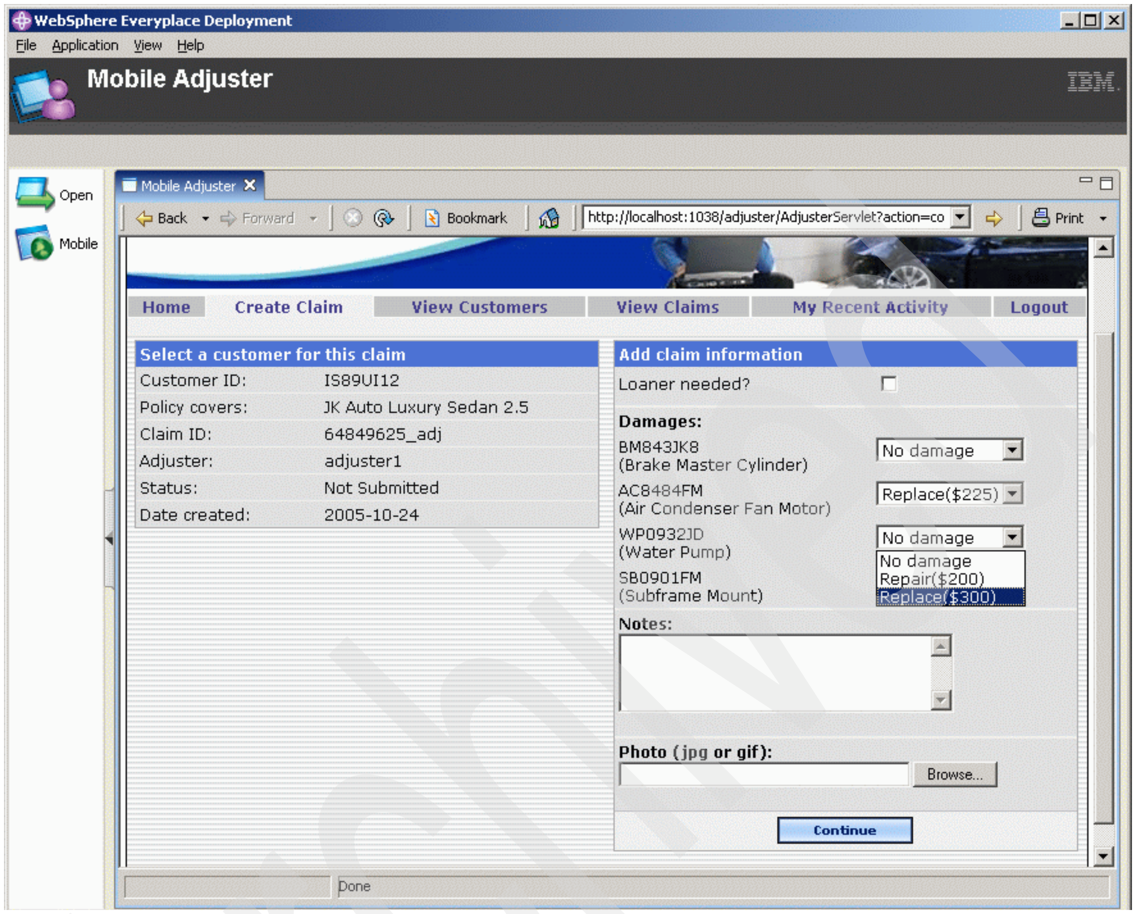
No image available.

Confirmation messages

No confirmation message has been received for this claim.

Figure 15-10 New claim information page

4. When the new claim information page appears, we entered the following:
   a. AC848FM (Air Condensor Fan Motor): Select **Replace ($225)**.
   b. WP0932JD (Water Pump): Select **Replace ($300)**.
   c. Optionally, you can attach an image (photo) to the claim form by clicking **Browse** and selecting the image.
   d. When done entering the claim data, the page should look like Figure 15-11 on page 420. Click **Continue**.



WebSphere Everyplace Deployment
File Application View Help
Mobile Adjuster
Open
Mobile
Back Forward Bookmark Print
http://localhost:1038/adjuster/AdjusterServlet?action=co
Home Create Claim View Customers View Claims My Recent Activity Logout
Browse...
Continue
Done

Select a customer for this claim	
Customer ID:	IS89UI12
Policy covers:	JK Auto Luxury Sedan 2.5
Claim ID:	64849625_adj
Adjuster:	adjuster1
Status:	Not Submitted
Date created:	2005-10-24

Add claim information

Loaner needed? ☐

Damages:

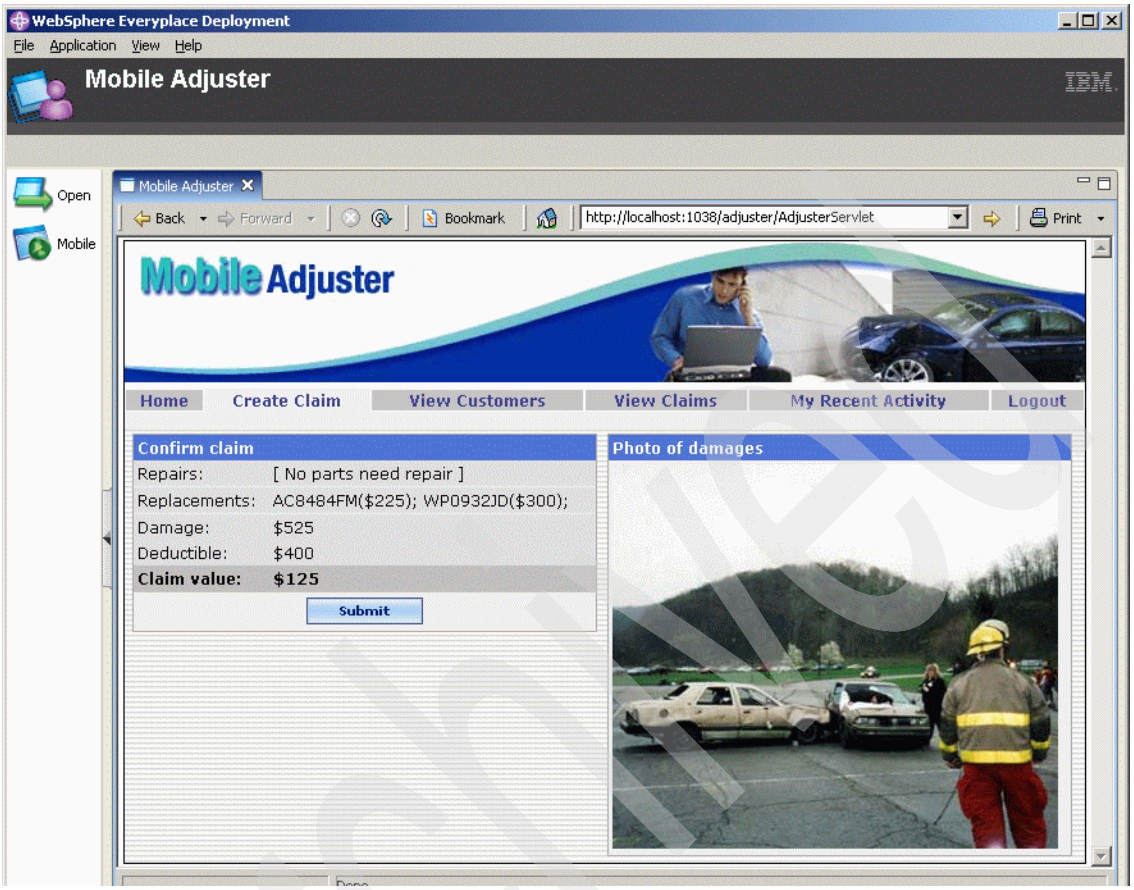
BM843JK8 (Brake Master Cylinder)	No damage
AC8484FM (Air Condenser Fan Motor)	Replace($225)
WP0932JD (Water Pump)	No damage
SB0901FM (Subframe Mount)	No damage / Repair($200) / Replace($300)

Notes:

Photo (jpg or gif):

Figure 15-11 Entering damage information to the claim form

5. When the Confirm claim page appears, it displays the damage, deductable, and claim value, as well as the photo of damages if present, as seen in Figure 15-12 on page 421.


WebSphere Everyplace Deployment
File Application View Help
Mobile Adjuster
Open
Mobile
Mobile Adjuster
Back Forward Bookmark Print
http://localhost:1038/adjuster/AdjusterServlet
Mobile Adjuster
Home Create Claim View Customers View Claims My Recent Activity Logout
Photo of damages
Submit

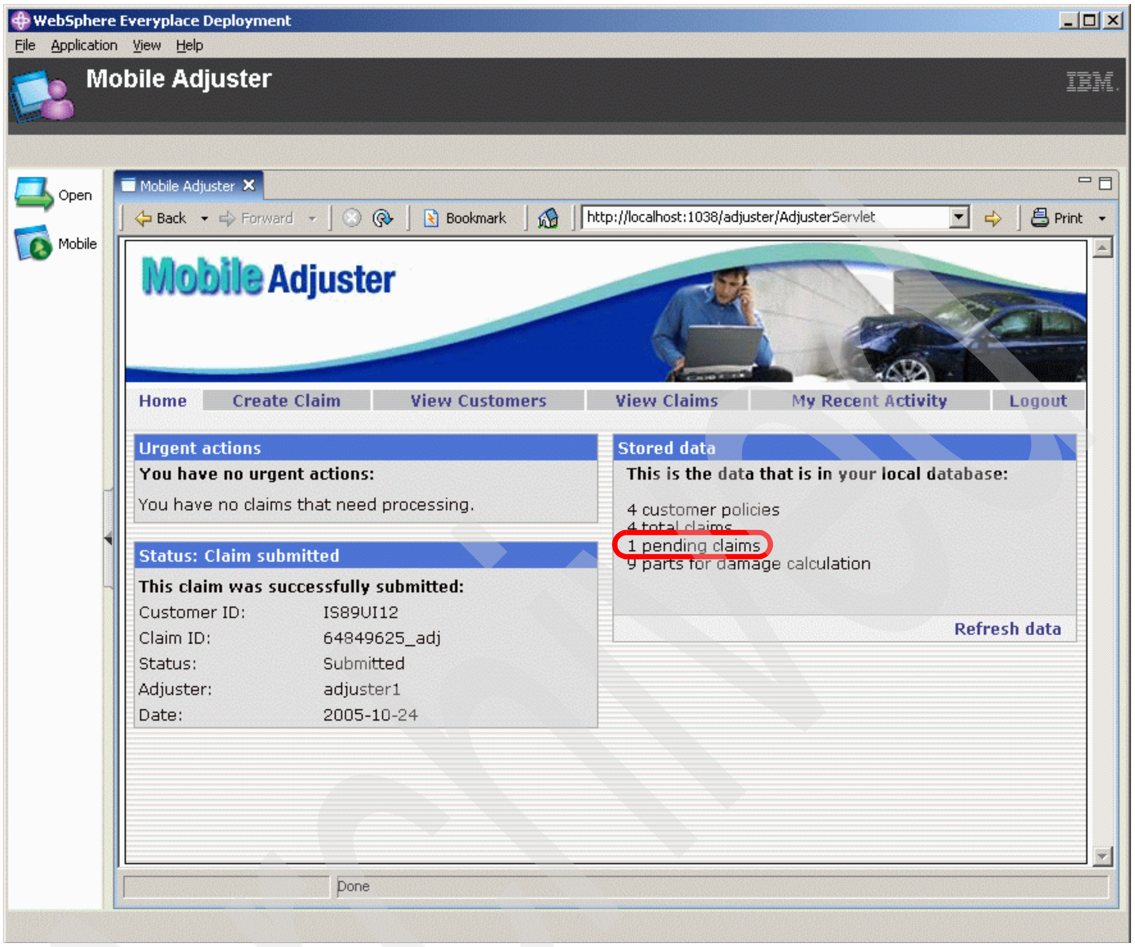
Confirm claim	
Repairs:	[No parts need repair]
Replacements:	AC8484FM($225); WP0932JD($300);
Damage:	$525
Deductible:	$400
Claim value:	**$125**

Figure 15-12 Confirm claim page

6. Click **Submit** to confirm claim data.

   The claim form data is stored in the local DB2e application database on the device, and a transactional message is sent to back-end system for automated processing since the claim value is less than $500.

   When the Mobile Adjuster home page appears, you will see that there is a new claim pending for synchronization (see Figure 15-13 on page 422).


WebSphere Everyplace Deployment
File Application View Help
Mobile Adjuster
IBM
Open
Mobile
Mobile Adjuster
Back Forward Bookmark Print
http://localhost:1038/adjuster/AdjusterServlet
Mobile Adjuster
Home Create Claim View Customers View Claims My Recent Activity Logout
Refresh data
Done

Urgent actions

You have no urgent actions:

You have no claims that need processing.

Status: Claim submitted

This claim was successfully submitted:

Customer ID:	IS89UI12
Claim ID:	64849625_adj
Status:	Submitted
Adjuster:	adjuster1
Date:	2005-10-24

Stored data

This is the data that is in your local database:

4 customer policies
4 total claims
1 pending claims
9 parts for damage calculation

Figure 15-13 New claim has been submitted and is waiting for synchronization

15.6 UC_6: Upload image

To avoid transmitting an excessive amount of data when the adjuster is on-site with the customer and potentially connected via a low bandwidth connection such as GPRS, the transactional messaging only sends essential data for claim processing. The text-based claim data stored in the local database and images are uploaded as separate tasks performed manually when the network connectivity is suitable.

To upload the vehicle accident images, do the following:

1. Start WebSphere Everyplace Deployment Client for Windows and Linux (client).
2. Click **Open**, and select **Mobile Adjuster Images**, as seen in Figure 15-14.

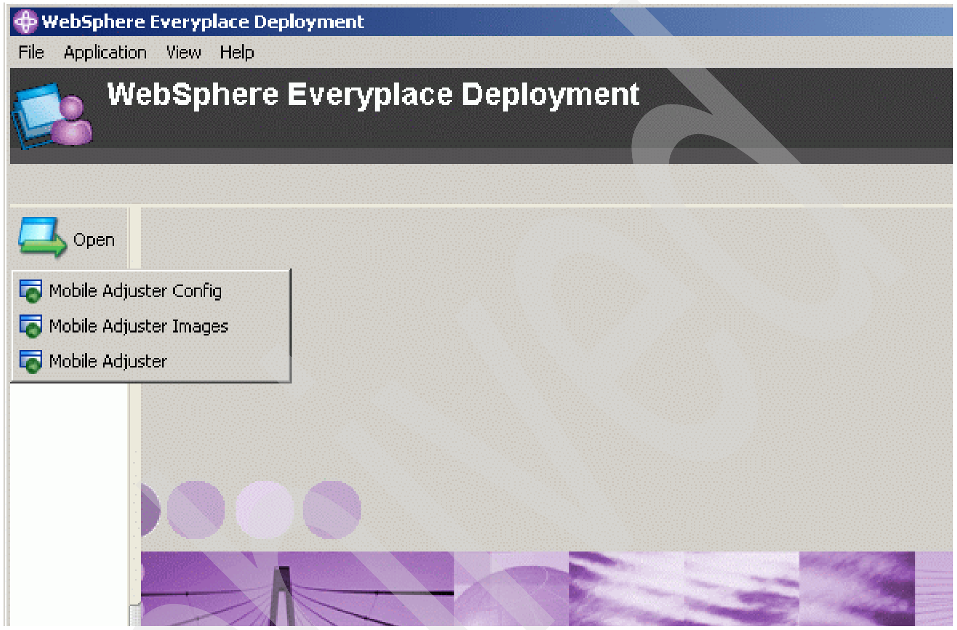

WebSphere Everyplace Deployment
File Application View Help
WebSphere Everyplace Deployment
Open
Mobile Adjuster Config
Mobile Adjuster Images
Mobile Adjuster

Figure 15-14 WebSphere Everyplace Deployment workplace

3. When the Mobile Adjuster Image Manager application View Claims With Images appears, click **Manage Images**.
4. All images stored on the client device are displayed. Click **Upload Image** (see Figure 15-15 on page 424) for each of the images you wish to upload.

   The Upload Image button is removed when the image is successfully uploaded to the server.

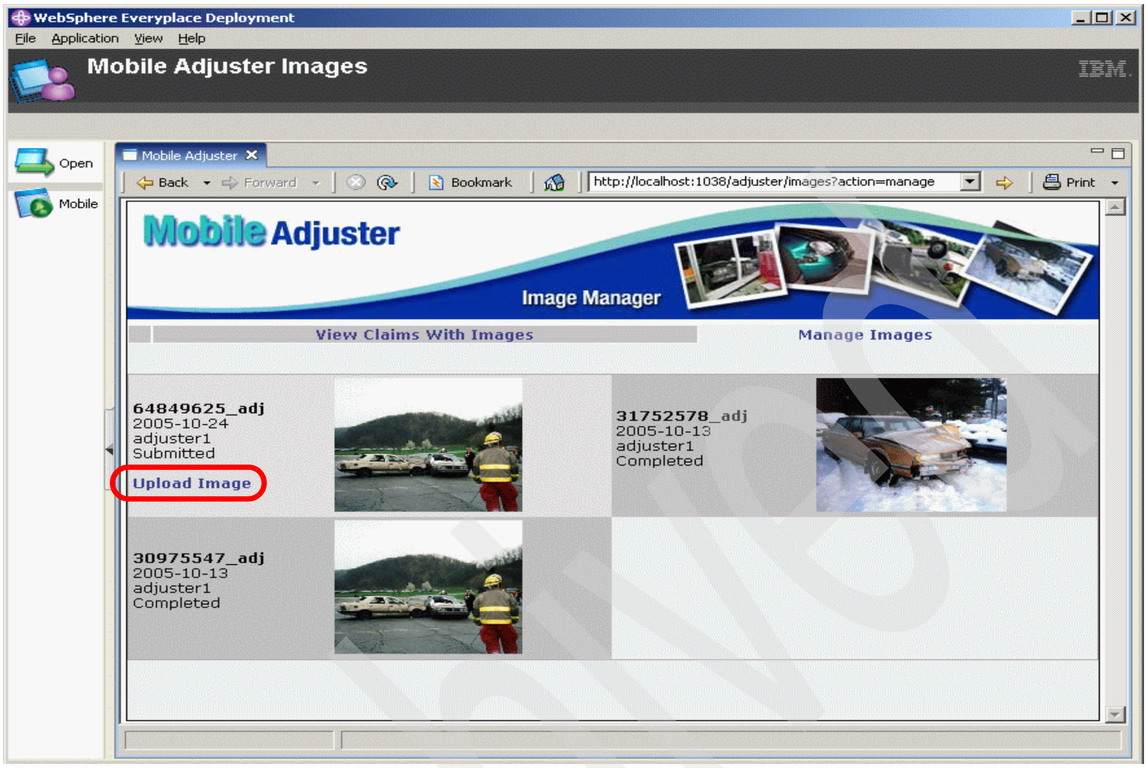


Figure 15-15 Upload Images

5. Close the Mobile Adjuster Image Manager application.

15.7 UC_7: View claims

Claims can be viewed by the agent in the Mobile Agent Web application or by the adjuster in the disconnectable Mobile Adjuster application.

View claims (agent - Mobile Agent Web application)

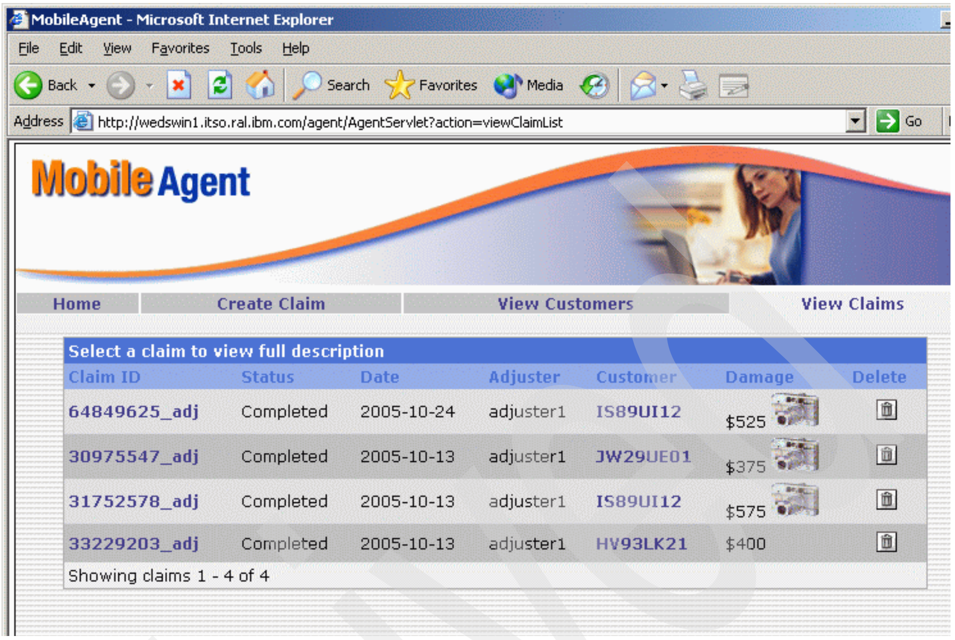
The agent can view a summary of all claims in the Mobile Agent online Web application as follows:

1. Enter the following URL in a Web browser to launch the Mobile Agent online Web application:

```
http://wedswin1.itso.ral.ibm.com/agent/AgentServlet
```

2. When the Mobile Agent home page appears, click **View Claims**.

   The resulting View Claims page will look like Figure 15-16 on page 425.


MobileAgent - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://wedswin1.itso.ral.ibm.com/agent/AgentServlet?action=viewClaimList
Go
Mobile Agent
Home
Create Claim
View Customers
View Claims

Select a claim to view full description

Claim ID	Status	Date	Adjuster	Customer	Damage	Delete
64849625_adj	Completed	2005-10-24	adjuster1	IS89UI12	$525	
30975547_adj	Completed	2005-10-13	adjuster1	JW29UE01	$375	
31752578_adj	Completed	2005-10-13	adjuster1	IS89UI12	$575	
33229203_adj	Completed	2005-10-13	adjuster1	HV93LK21	$400	

Showing claims 1 - 4 of 4

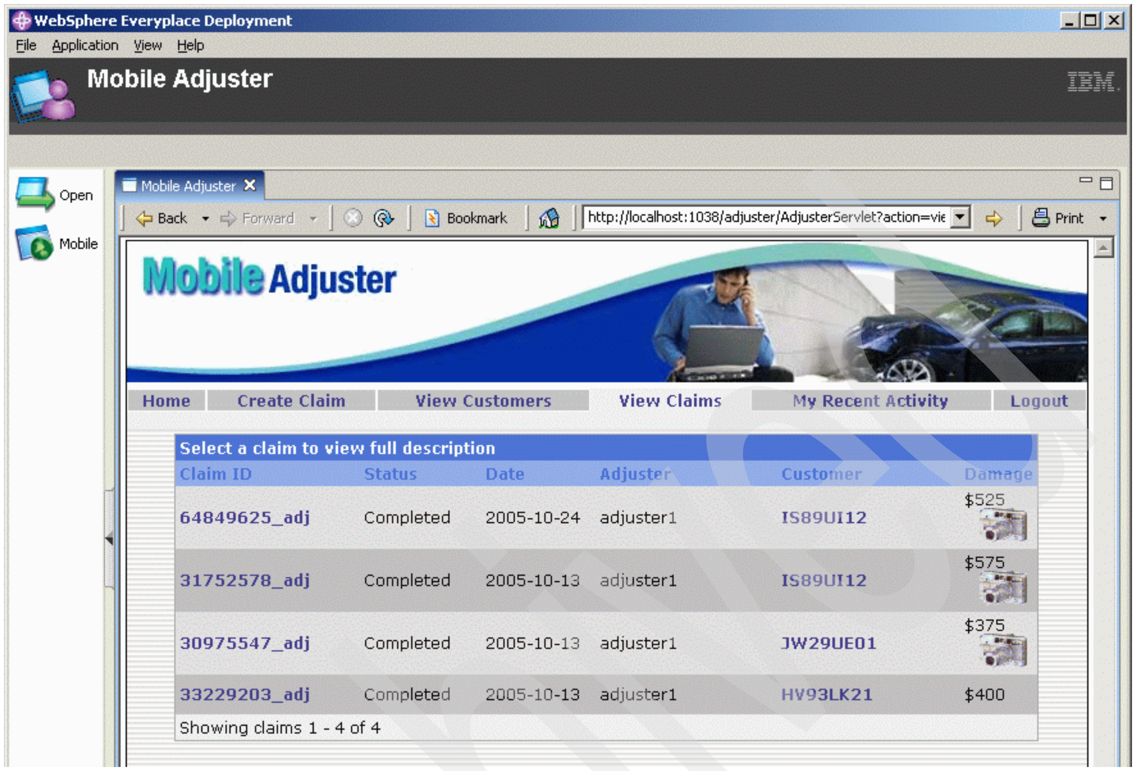
Figure 15-16 View Claims - Online Mobile Agent

View claims (adjuster - disconnetable Mobile Adjuster)

Alternatively, the adjuster can view a summary of all claims in the disconnectable Mobile Adjuster application.

1. Start WebSphere Everyplace Deployment Client for Windows and Linux (client).
2. Click **Open** and select **Mobile Adjuster**.
3. When the Login dialog appears, enter the user name and password (for example, adjuster1), and then click **Login**.
4. From the Home page, click **Refresh Data** to ensure you have synchronzied the latest claims data to be viewed.
5. Click **View Claims**.

   The resulting page should look like Figure 15-17 on page 426.


WebSphere Everyplace Deployment
File Application View Help
Mobile Adjuster
IBM
Open
Mobile
Mobile Adjuster
Back Forward Bookmark Print
http://localhost:1038/adjuster/AdjusterServlet?action=vie
Mobile Adjuster
Home Create Claim View Customers View Claims My Recent Activity Logout

Select a claim to view full description

Claim ID	Status	Date	Adjuster	Customer	Damage
64849625_adj	Completed	2005-10-24	adjuster1	IS89UI12	$525
31752578_adj	Completed	2005-10-13	adjuster1	IS89UI12	$575
30975547_adj	Completed	2005-10-13	adjuster1	JW29UE01	$375
33229203_adj	Completed	2005-10-13	adjuster1	HV93LK21	$400

Showing claims 1 - 4 of 4

Figure 15-17 View Claims - Disconnectable Mobile Adjuster

15.8 UC_8: View customer profile

The customer profile can be viewed by agents in the online Mobile Agent Web application or by adjusters in the disconnectable Mobile Adjuster application.

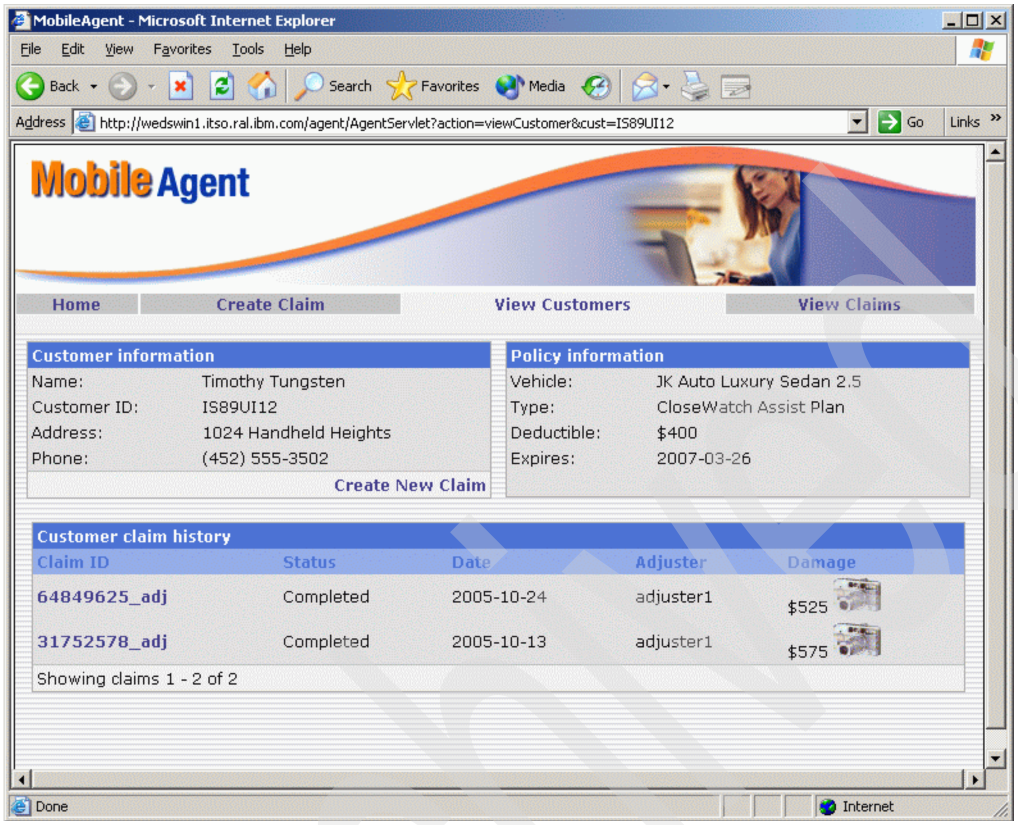
View customer profile (agent - Mobile Agent Web application)

The agent can view customer profile information and claims using the online Mobile Agent Web application, as follows:

1. Enter the following URL in a Web browser to launch the Mobile Agent online Web application:

   ```
   http://wedswin1.itso.ral.ibm.com/agent/AgentServlet
   ```

2. When the Mobile Agent home page appears, click **View Customers**.
3. Click the desired customer to display the customer profile information. The resulting page should look like Figure 15-18 on page 427.


MobileAgent - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://wedswin1.itso.ral.ibm.com/agent/AgentServlet?action=viewCustomer&cust=IS89UI12
Go Links
Mobile Agent
Home
Create Claim
View Customers
View Claims
Done
Internet

Customer information	
Name:	Timothy Tungsten
Customer ID:	IS89UI12
Address:	1024 Handheld Heights
Phone:	(452) 555-3502
	Create New Claim

Policy information	
Vehicle:	JK Auto Luxury Sedan 2.5
Type:	CloseWatch Assist Plan
Deductible:	$400
Expires:	2007-03-26

Customer claim history

Claim ID	Status	Date	Adjuster	Damage
64849625_adj	Completed	2005-10-24	adjuster1	$525
31752578_adj	Completed	2005-10-13	adjuster1	$575

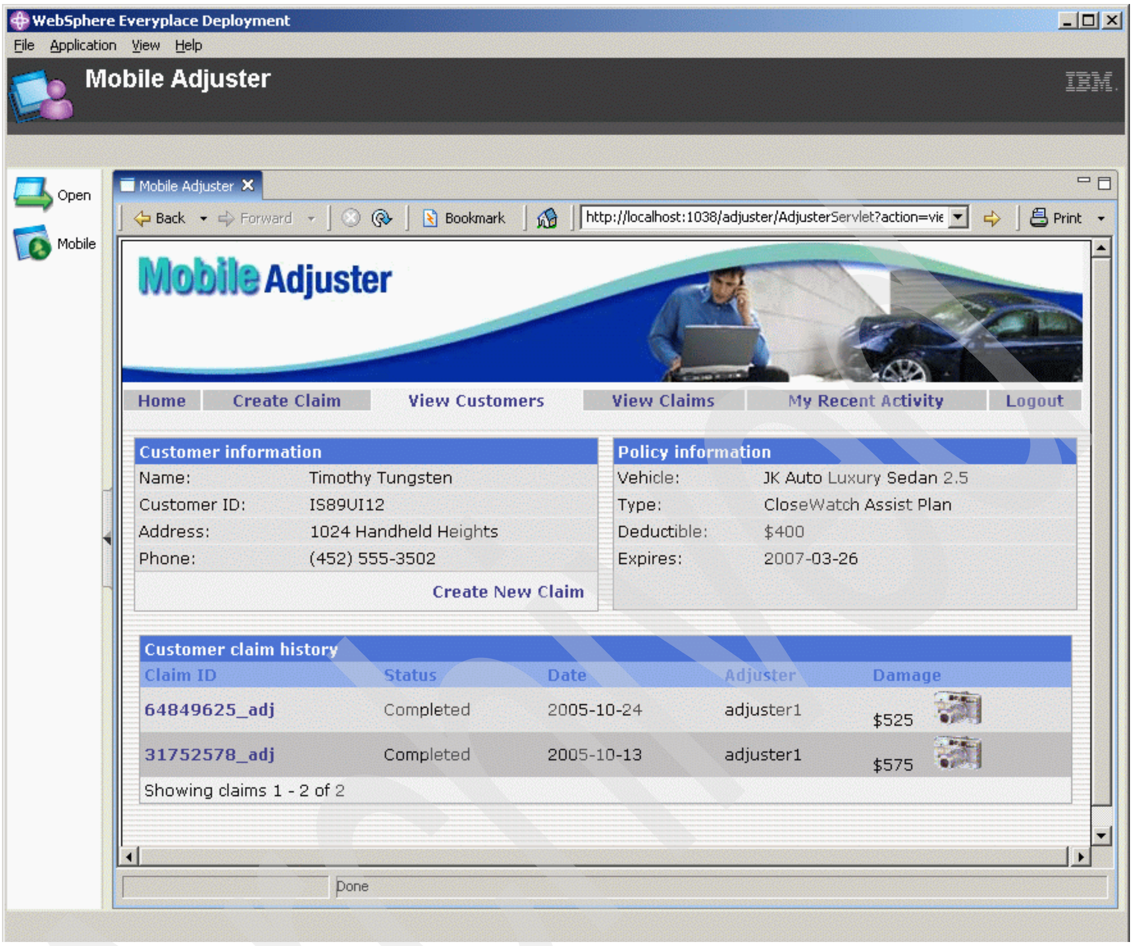
Showing claims 1 - 2 of 2

Figure 15-18 Customer profile - Online Mobile Agent

View customer profile (adjuster - Mobile Adjuster)

Alternatively, the adjuster can view customer profiles and claims from the Mobile Adjuster application:

1. Start WebSphere Everyplace Deployment Client for Windows and Linux (client).
2. Click **Open** and select **Mobile Adjuster**.
3. When the Login dialog appears, enter the user name and password (for example, adjuster1), and then click **Login**.
4. From the Home page, click **Refresh Data** to ensure you have synchronzied the latest customer profile information to be viewed.
5. Click **View Customers**.
6. Select the desired customer you wish to view. The resulting page should look like Figure 15-19 on page 428.


WebSphere Everyplace Deployment
File Application View Help
Mobile Adjuster
Open
Mobile
Mobile Adjuster
Back Forward Bookmark
http://localhost:1038/adjuster/AdjusterServlet?action=vie
Print
Mobile Adjuster
Home Create Claim View Customers View Claims My Recent Activity Logout
Create New Claim
Done

Customer information	
Name:	Timothy Tungsten
Customer ID:	IS89UI12
Address:	1024 Handheld Heights
Phone:	(452) 555-3502

Policy information	
Vehicle:	JK Auto Luxury Sedan 2.5
Type:	CloseWatch Assist Plan
Deductible:	$400
Expires:	2007-03-26

Customer claim history

Claim ID	Status	Date	Adjuster	Damage
64849625_adj	Completed	2005-10-24	adjuster1	$525
31752578_adj	Completed	2005-10-13	adjuster1	$575

Showing claims 1 - 2 of 2

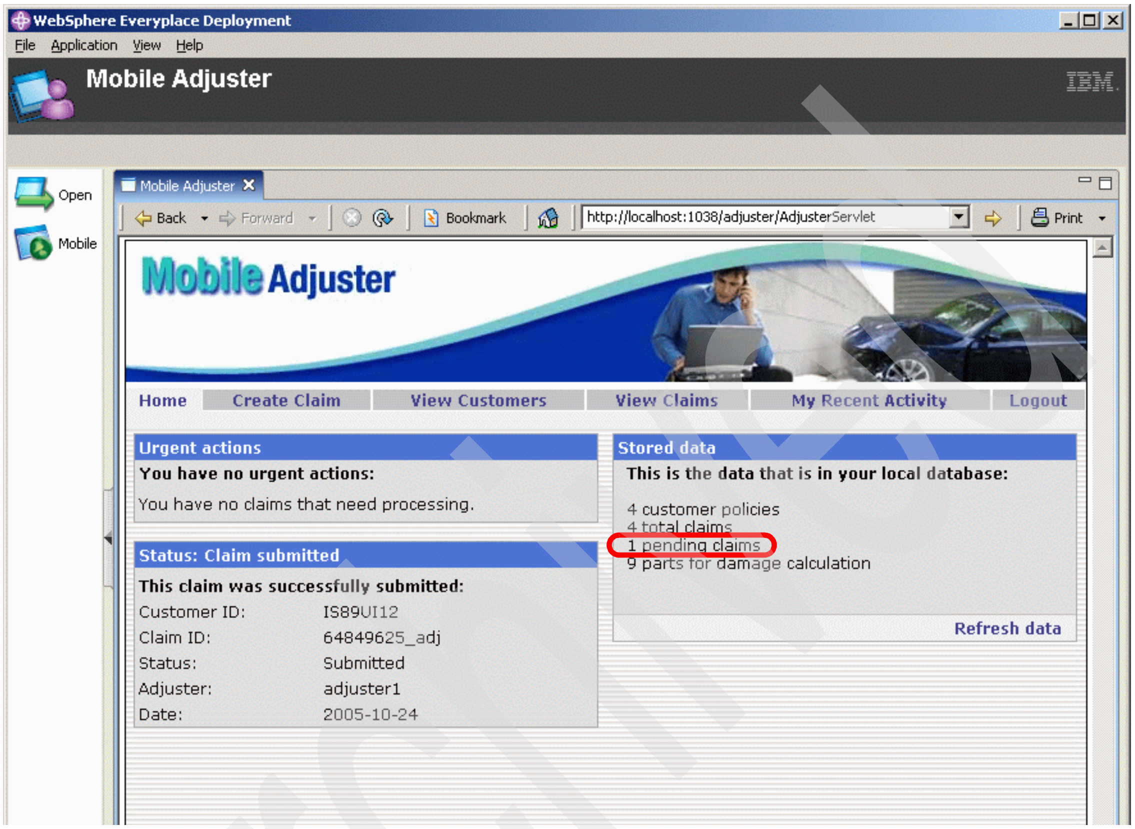
Figure 15-19 Customer profile - Disconnectable Mobile Adjuster

15.9 UC_9: Sync claims data

Claims prepared by the adjuster in the disconnectable Mobile Adjuster application need to be synchronized with the enterprise back-end database.

1. Start WebSphere Everyplace Deployment Client for Windows and Linux (client).
2. Click **Open** and select **Mobile Adjuster**.
3. When the Login dialog appears, enter the user name and password (for example, adjuster1), and then click **Login**.

4. When the home page appears, notice the pending claims (see Figure 15-20). Click **Refresh Data** to synchronize the claims data.


WebSphere Everyplace Deployment

File Application View Help

Mobile Adjuster

Open

Mobile

Mobile Adjuster

Back Forward Bookmark http://localhost:1038/adjuster/AdjusterServlet Print

Mobile Adjuster

Home | Create Claim | View Customers | View Claims | My Recent Activity | Logout

Urgent actions

You have no urgent actions:

You have no claims that need processing.

Status: Claim submitted

This claim was successfully submitted:

Customer ID:	IS89UI12
Claim ID:	64849625_adj
Status:	Submitted
Adjuster:	adjuster1
Date:	2005-10-24

Stored data

This is the data that is in your local database:

4 customer policies
4 total claims
1 pending claims
9 parts for damage calculation

Refresh data

Figure 15-20 Home page of Mobile Adjuster application with claim to be synchronized

5. You should see the message dialog `Data refreshed successfully`. Click **OK**.

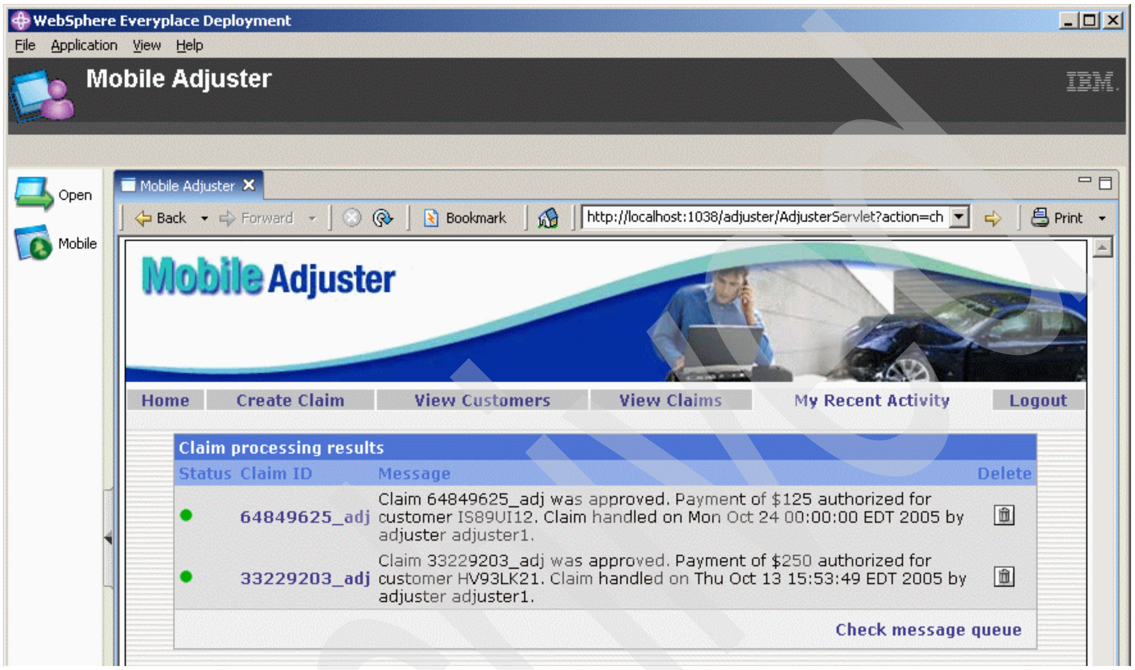
   The Home page should now show no pending claims.

15.10 UC_10: Refresh claim status

The adjuster can check for recent messages delivered regarding claim status.

1. Start WebSphere Everyplace Deployment Client for Windows and Linux (client).
2. Click **Open** and select **Mobile Adjuster**.
3. When the Login dialog appears, enter the User Name and Password (for example, adjuster1), and then click **Login**.

4. When the home page appears, click **My Recent Activity**.
5. The messages should be displayed. The adjuster can also click **Check message queue**, as seen in Figure 15-21, to retrieve new messages to be displayed.



Claim processing results

Status	Claim ID	Message	Delete
●	64849625_adj	Claim 64849625_adj was approved. Payment of $125 authorized for customer IS89UI12. Claim handled on Mon Oct 24 00:00:00 EDT 2005 by adjuster adjuster1.	
●	33229203_adj	Claim 33229203_adj was approved. Payment of $250 authorized for customer HV93LK21. Claim handled on Thu Oct 13 15:53:49 EDT 2005 by adjuster adjuster1.	

Check message queue

Figure 15-21 My Recent Activity screen with Check message queue button

Part 4

Appendixes

A

IBM software product descriptions

The objective of this appendix is to provide a brief product description for each IBM software product used in the Product mappings of this book. In some cases, the products are available as standalone products, and in other cases the software components are bundled in solution offerings.

The appendix is organized into the following sections:

- IBM DB2 Everyplace
- IBM WebSphere MQ Everyplace
- IBM WebSphere Everyplace Micro Environment
- IBM Workplace Client Technology, Micro Edition
- IBM WebSphere Everyplace Mobile Portal Enable
- IBM WebSphere Everyplace Deployment
- IBM WebSphere Everyplace Access
- IBM WebSphere Everyplace Connection Manager
- IBM WebSphere Everyplace Device Manager
- IBM WebSphere Voice Server
- IBM WebSphere Voice Application Access
- IBM WebSphere Voice Response for AIX
- IBM WebSphere Application Server
- IBM WebSphere Portal
- IBM DB2 Universal Database

- IBM WebSphere Everyplace Client
- IBM Lotus Domino
- IBM Lotus Sametime
- IBM Workplace Collaboration Services

IBM DB2 Everyplace

IBM DB2 Everyplace (DB2e) features a small footprint relational database and high performance data synchronization solution. DB2e enables enterprise applications and data to be securely extended to a wide range of client devices such as laptops, personal digital assistants (PDAs), smart phones, and other embedded mobile devices. With DB2e the mobile workforce in industries such as health care, telecommunications, retail, distribution, transportation, insurance, and hospitality can easily access the information they need to perform their work from any location, at any time, right from the palms of their hands.

DB2e can be used as a local independent database of the mobile device or query information about remote servers when a connection is available.

In a DB2e architecture, the Sync Server sits on the middleware tier. Essentially, its purpose is to serve as the intermediary between the source database and the client devices.

Thanks to the JDBC industry standard, DB2e can support a wide variety of enterprise databases (DB2 UDB, Oracle, etc.).

IBM DB2 Everyplace integrates with a wide range of products such as IBM WebSphere Everyplace Deployment V6.0 and IBM WebSphere Everyplace Access V5.1.

Note: More information about IBM DB2 Everyplace (DB2e) can be found at:

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

IBM WebSphere MQ Everyplace

IBM WebSphere MQ Everyplace (MQe) is a member of the IBM WebSphere MQ family of 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 support mobility and the requirements that arise from the use of fragile communication networks. It maintains the standard WebSphere MQ quality of service, providing once-only assured delivery, and exchanges messages with other family members. Since many MQe applications run outside the protection of an Internet firewall, it also provides sophisticated security capabilities.

MQe supports the following:

- Broad mobile device support: Supports a wide range of devices with a small, customizable footprint. Offers a choice of languages, APIs, and environments including Java, C, JMS, and J2ME™.
- Robust mobile integration: Once-only messaging so transactions are not lost or duplicated between mobile applications. Peer-to-peer, synchronous, and asynchronous messaging. Rich encryption, non-repudiation, and authentication features.
- Compliments synchronization and portal capabilities of WebSphere Everyplace Access and WebSphere Everyplace Deployment by providing robust message delivery to address the problem of intermittent network connectivity.
- Ideal when combined with WebSphere Everyplace Connection Manager to create robust and highly flexible mobile and wireless solutions (seamless roaming, broad network coverage, network abstraction to TCP/IP layer, compression and high transport security).
- It also extends other WebSphere Business Integration offerings and transports to the mobile and wireless environment.

IBM WebSphere MQ Everyplace is included as a component of IBM WebSphere Everyplace Deployment V6.0.

Note: More information about IBM WebSphere MQ Everyplace can be found at:

http://www.ibm.com/software/integration/wmqe

IBM WebSphere Everyplace Micro Environment

IBM WebSphere Everyplace Micro Environment (WEME) is a Java-powered runtime environment that provides the foundation for the deployment of e-business applications to small mobile devices. The WebSphere Everyplace Micro Environment has been tested and certified to meet the J2ME specifications as laid out by the Java Community Process. WebSphere Everyplace Micro Environment supports:

- Connected, Limited Device Configuration (CLDC 1.0 and 1.1) and Mobile Information Device Profile (MIDP 2.0)
- Connected Device Configuration (CDC 1.0_01), Foundation Profile, and Personal Profile

The use of industry standards and middleware enables devices to be connected to many existing e-business applications. By testing middleware and server connectivity, WebSphere Everyplace Micro Environment combines the convenience of mobile devices with the power of e-business. When used with other middleware products, many existing enterprise applications can be extended to server-managed mobile and pervasive devices:

- Web services (SOAP) support can help enable access to applications across a wide variety of wireless and wire-line networks.
- Testing with assured messaging software (MQe, JMS) can provide access to assured messaging and financial transactions.
- When used with DB2e, the advanced data management capabilities of DB2 can also be leveraged.
- Applications, middleware, and runtimes can be server-managed when used in conjunction with the IBM Service Management Framework.

Note: More information about IBM WebSphere Everyplace Micro Environment (WEME) can be found at:

http://www.ibm.com/software/wireless/weme

IBM Workplace Client Technology, Micro Edition

IBM Workplace Client Technology, Micro Edition (WCTME) V5.7.x, provides an integrated platform for the extension of existing enterprise applications to server-managed client devices, such as desktops, laptop, personal digital assistants (PDAs), and other mobile and pervasive devices.

The integrated package combines the tools (WebSphere Studio Device Developer and Micro Environment Toolkit for WebSphere Studio), runtimes (WebSphere Everyplace Micro Environment, Service Management Framework, and WebSphere Everyplace Custom Environment), and middleware (DB2e, MQe, Web Services) for building, testing, and deploying server-managed client software to pervasive devices.

The use of industry standards and middleware enables devices to be connected to many existing e-business applications. By testing middleware and server connectivity, WebSphere Everyplace Micro Environment combines the convenience of mobile devices with the power of e-business. When used with other middleware products, many existing enterprise applications can be extended to server-managed mobile and pervasive devices. Web services (SOAP) support can help enable access to applications across a wide variety of wireless and wire-line networks.

Testing with assured messaging software (MQe, JMS) can provide access to assured messaging and financial transactions.

When used with DB2e, the advanced data management capabilities of DB2 can also be leveraged. Applications, middleware, and runtimes can be server-managed when used in conjunction with the IBM Service Management Framework.

IBM has combined the portability of Java technology with open and industry standards to deliver a comprehensive platform for extending many existing business processes onto millions of small devices.

IBM Workplace Client Technology, Micro Edition is a supported client of IBM WebSphere Everyplace Deployment V6.0 and IBM WebSphere Everyplace Access V5.1.

Note: More information about IBM Workplace Client Technology, Micro Edition (WCTME) can be found at:

http://www.ibm.com/software/wireless/wctme

IBM WebSphere Everyplace Mobile Portal Enable

IBM WebSphere Everyplace Mobile Portal Enable V5.1 (WEMP) builds on the capabilities of WebSphere Portal to deliver personalized, aggregated content to mobile users. The content is authored once and Mobile Portal renders it appropriately for the various mobile devices. When we talk about content, we are referring to the elements that are presented by a portlet or a Web page to convey information to mobile users. In the early days of the Internet, Web pages were purely textual. Over time, the content of Web pages has evolved significantly to incorporate images, multimedia, and interactive elements to provide a much richer user experience. HTML, the markup of choice for representing content on PC-based browsers, has provided the infrastructure for developers to deliver this rich content to users. Mobile Portal uses XDIME, a device-independent mark-up language, to provide similar support for delivering rich content to mobile devices.

IBM WebSphere Everyplace Mobile Portal Enable V5.1 is an integrated server-side solution that enables:

- Multi-channel access of portal content
- Write once, render on multiple devices
- Navigation, personalized experience
- User registration and profile management from land-line Web sites

- Security, access control, single sign-on
- Aggregates any mix of applications into a unified display
- Defines a Model View Control (MVC) structure, modes, and view states
- Separates portlet development from page layout and branding
- Tooling support for application development-markup/layout editors, samples, APIs, documentation
- Templates for rapid site development and enforcement of style guidelines
- Includes XSLTs to translate existing markup to new and future XHTML devices

Note: More information about IBM WebSphere Everyplace Mobile Portal Enable V5.1 can be found at:

http://www.ibm.com/software/pervasive/ws_everyplace_mobile_portal_enable

IBM WebSphere Everyplace Deployment

IBM WebSphere Everyplace Deployment V6.0 (WED) is an end-to-end platform that provides the Java-based server-managed client platform, the tooling, and the corresponding server-side components required to build, deploy, and maintain sophisticated solutions that can adapt to device and network conditions. IBM WebSphere Everyplace Deployment provides the capability to extend existing enterprise applications to the edge, for a wide range of online and disconnectable devices such as desktops, laptops, PDAs, and mobile phones.

We have organized the overview of IBM WebSphere Everyplace Deployment into the following sections:

- Software component packaging
- Key capabilities and services

Software component packaging

The IBM WebSphere Everyplace Deployment (WED) product is packaged into three distinct sets of software components:

- IBM WebSphere Everyplace Deployment (server)
- IBM WebSphere Everyplace Deployment for Windows/Linux (client)
- IBM WebSphere Everyplace Client Toolkit (development tooling)

IBM WebSphere Everyplace Deployment

IBM WebSphere Everyplace Deployment V6.0 is the server platform of IBM WebSphere Everyplace Deployment (WED). The WED V6.0 server platform is supported on the 32-bit Microsoft Windows 2003 Standard or Enterprise Edition with or without Service Pack 1.

The WED server requires the following software:

- IBM WebSphere Application Server V6.0.2 with APAR PK104
- IBM HTTP Server V6.0.2
- IBM DB2 Universal Database V8.2 with FP9 (or FP9a)

It can be optionally configured to use Microsoft Active Directory included with Microsoft Windows 2003 Server with SP1.

The WED server platform includes the following key server-side services:

- Core services
- IBM DB2 Everyplace Sync Server V8.2.1
- Device Manager server V1.8
- MQ Everyplace Server V2.0.1.8
- Web Services (provided by WebSphere Application Server)

The WED server platform supports the following client platforms:

- Win32/Linux desktop and laptop clients:
  - IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (packaged with IBM WebSphere Everyplace Deployment V6.0)
  - IBM Workplace Client Technology Micro Edition Enterprise Offering V5.8.1 (WCTME-EO) (packaged and sold separately)
- Palm and Windows Mobile 2003 2nd Edition PDA clients:
  - IBM Workplace Client Technology Micro Edition V5.7.1 (WCTME) (packaged and sold separately)
  - IBM Workplace Client Technology Micro Edition V5.7.2 FP1 (WCTME) (packaged and sold separately)

IBM WebSphere Everyplace Deployment for Windows/Linux

IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux is supported by client devices running Microsoft Windows XP (SP1 or SP2) and Red Hat Enterprise Linux Workstation, Version 3.0 - Update 3.

The WED client platform includes the following key services:

- Device Agent V1.8 (OSGI based): Install and maintain software and client device.

- DB2 Everyplace or Cloudscape: Synchronize relational data.
- MQ Everyplace Client V2.0.1.8: Send and receive secure transactions.
- Web Services: Consume and publish Web Services using an SOA industry standard.
- Eclipse: Web browser or rich user interface provided by the open source platform Eclipse technology.

IBM WebSphere Everyplace Client Toolkit

The IBM WebSphere Everyplace Client Toolkit provides a complete, integrated set of tools that allows you to develop, debug, test, package and deploy client applications that use the client services. These extensions leverage the enterprise developer community assets by using the same Java 2 Enterprise Edition (J2EE) programming model for application development.

The IBM WebSphere Everyplace Deployment V6.0 product includes the WebSphere Everyplace Client Toolkit V6.0 used to develop client-side applications to be deployed to IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux (clients). The WebSphere Everyplace Client Toolkit can be installed on one of the following IBM Rational Software Developer V6 products:

- IBM Rational Web Developer V6.0.0.1 (RWD) or later
- IBM Rational Application Developer V6.0.0.1 (RAD) or later
- IBM Rational Software Architect V6.0.0.1 (RSA) or later

The IBM Rational Software Developer products listed are in order of least to most capable. For example, RSA includes all enterprise development capabilities of RAD, but also includes modeling and design tooling used by architects.

The IBM WebSphere Everyplace Client Toolkit provides the tools necessary to create and test OSGi bundles, Web applications, and embedded transaction applications for use on the WebSphere Everyplace Deployment platform. The WebSphere Everyplace Client toolkit provides wizards that enable developers to create a Client Services project to develop client applications, without requiring them to become OSGi bundle internal experts.

We have listed some of the key features provided by the WECT:

- Client Services Embedded Transaction Project - Develop and deploy EJBs.
- Client Services Fragment Project - Develop fragments.
- Client Services Project - Develop Eclipse-based rich client user interfaces applications, messaging applications, database applications and application components (plug-in/bundle).
- Client Services Web Project - Develop and deploy Web applications.

- Convert Project to Client Service Project - Convert a Java project to a Client Services project.
- Migrate Extension Services Project - Migrate a WSDD 5.7 Extension Services project to a Client Services project.
- Platform Builder Project - Build a custom client platform.
- Mobile Web Services - Develop a Web Services client and/or provider and support security.

The WECT provides the ability to develop the following types of client applications:

- Eclipse Rich Client Platform applications

> **Note:** Within the patterns chapters, this capability is known as *C19 Disconnectable applications using the local Eclipse based rich user interface*.

- Web applications

> **Note:** Within the patterns chapters, this capability is known as *C15 Disconnectable application with local Web browser user interface*.

- Embedded Transaction applications
- Messaging applications

> **Note:** Within the patterns chapters, this capability is known as *C17 Disconnectable applications using transactional messaging*.

- Database applications

> **Note:** Within the patterns chapters, this capability is known as *C16 Disconnectable applications using relational sync (local data store, sync. with server)*.

- Web Services applications

> **Note:** Within the patterns chapters, this capability is known as *C18 Disconnectable applications using Web services (publish/consume Web services)*.

Key capabilities and services

Figure A-1 highlights the key capabilities by providing standards-based client services that can be used to extend the enterprise:

- Managed client services
- Access services
- Interaction services
- Platform Management services

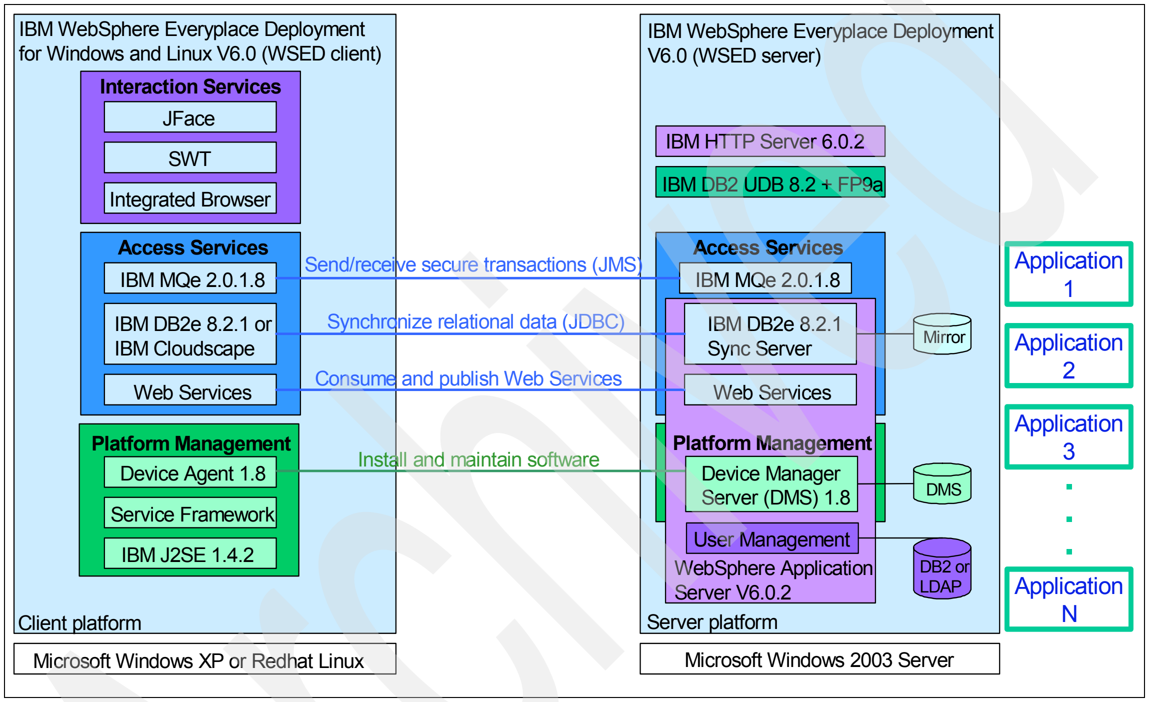

IBM WebSphere Everyplace Deployment for Windows and Linux V6.0 (WSED client)
Interaction Services
JFace
SWT
Integrated Browser
Access Services
IBM MQe 2.0.1.8
IBM DB2e 8.2.1 or IBM Cloudscape
Web Services
Platform Management
Device Agent 1.8
Service Framework
IBM J2SE 1.4.2
Client platform
Microsoft Windows XP or Redhat Linux
Send/receive secure transactions (JMS)
Synchronize relational data (JDBC)
Consume and publish Web Services
Install and maintain software
IBM WebSphere Everyplace Deployment V6.0 (WSED server)
IBM HTTP Server 6.0.2
IBM DB2 UDB 8.2 + FP9a
Access Services
IBM MQe 2.0.1.8
IBM DB2e 8.2.1 Sync Server
Mirror
Web Services
Platform Management
Device Manager Server (DMS) 1.8
DMS
User Management
WebSphere Application Server V6.0.2
DB2 or LDAP
Server platform
Microsoft Windows 2003 Server
Application 1
Application 2
Application 3
Application N

Figure A-1 IBM WebSphere Everyplace Deployment V6.0 end-to-end solution

Managed client services

The WED client platform includes the IBM Java 2 Standard Edition (J2SE) 1.4.2 JVM (service release 2) as the base Java runtime environment for Windows XP and RedHat Linux.

The client platform provides a Service Framework that implements the OSGi R3 framework specification and provides a service-oriented architecture on top of the JVM. The OSGi framework specification is provided by the OSGi Alliance. The OSGi Alliance's mission is to specify, create, advance, and promote wide industry adoption of an open service delivery and management platform.

Incorporating the OSGi standard into the client platform provides four very important capabilities:

- Multiple applications and components to share a single JVM, thus saving valuable resources on the client (rather than multiple JVM instances).
- Applications can share services and packages (reduce resources).
- Separates service interface from service implementation and provides publish, find, and bind operations in support of a service-oriented architecture. This capability enables integration of business applications on the same device.
- Enables dynamic life-cycle management without a JVM restart so components can be updated without impacting other unrelated components that are running at the same time.

The Eclipse RCP framework is built on the Service Framework, which provides the Eclipse RCP with powerful capabilities, such as the ability to dynamically load and unload components without restarting the Eclipse RCP framework and robust life-cycle management of components.

The client platform also provides optional OSGi services, such as UserAdmin, LogService, Configuration Management, and more.

Access services

Access services provide a familiar programming model for J2EE developers so they can reuse their skills and software components to develop applications that run on clients. Additionally, Access services enable client applications to support disconnectable applications. Access services also enable you to move key components of your application to the client platform through the use of standard APIs.

The client platform provides an embedded Web container to run J2EE Web applications that support either the Servlet 2.3 and JSP 1.2 specifications, or the Servlet 2.4 and JSP 2.0 specifications. The Web container enables you to move your Web applications from the server to clients to preserve the existing browser user interface, leverage your existing Web components, and provide a richer user experience through support of local and disconnectable applications.

The client platform also provides an embedded Transaction Container to run J2EE Enterprise Java Beans (EJBs) that conform to any of the following specifications: 1.1 and 2.0 Stateless Session Beans, Container Managed Persistence (CMP) Entity Beans, and Bean Managed Persistence (BMP) Entity Beans. This container enables you to move your business logic from the server to clients so you can leverage your existing beans to make business logic available to client applications, including Web applications, and support local and offline operations. These business logic components are referred to as Embedded Transaction applications.

There are two key services that support disconnectable applications:

- JDBC API with IBM DB2 Everyplace (DB2e) or IBM Cloudscape
- JMS API with IBM WebSphere MQ Everyplace (MQe)

JDBC API with IBM DB2 Everyplace (DB2e) or IBM Cloudscape

You can use the open standard Java Database Connectivity (JDBC) API with IBM DB2 Everyplace (DB2e) or IBM Cloudscape as a local SQL database when more advanced data manipulations are required than can be supported by placing data in a local file store. These databases can periodically synchronize with enterprise databases to capture data on the client for use by the client application when the user is disconnected. DB2e databases can also protect local data through data encryption. DB2e is an extremely small footprint relational database (200-300 KB). It is especially suitable for embedded devices, where large databases and sophisticated queries are not normally required, but can also be used on larger devices. DB2e provides transaction support covering updates to multiple tables within a single transaction, encrypted tables, and zero client administration.

IBM Cloudscape is a 100 percent pure Java relational database, providing SQL-92, partial SQL-99, and SQLJ support, indexes, triggers, transactions, encryption, and the standard features that one expects of a relational database. Since IBM Cloudscape contains a larger number of features, it is approximately 2 MB in size. Therefore, IBM Cloudscape might not be suitable for smaller, resource-constrained devices.

On the WED server the DB2 SyncServer is provided by DB2 Everyplace. The DB2 SyncServer provides the ability to synchronize data from the mobile device to other data sources in your enterprise.

JMS API with IBM WebSphere MQ Everyplace (MQe)

You can also use the open standard Java Message Service (JMS) API with IBM WebSphere MQ Everyplace (MQe) to send and receive messages. MQe provides once-only, assured messaging and supports offline operations with local message queues that hold messages when the device is offline and then sends these queued messages to Enterprise applications when the device is back online. Similarly, messages destined for client applications are held in server-side message queues and then sent to the client applications when the device is back online. MQe encrypts messages to protect content over the network to secure e-business transactions.

Other access services

The WED client platform also includes three additional access services we would like to highlight.

- *Web Services:* The client platform provides online client applications with support to consume or provide for Web Services in a secure manner. This provides users access to a broad range of business data and information.
- *MicroBroker technical preview:* The client platform also supports a technical preview of the MicroBroker, which is suitable for applications that require messaging, notification, and event services.
- *SyncML4J:* The SyncML4J (SyncML for Java) toolkit enables you to develop data synchronization and device management client applications based on the Open Mobile Alliance (OMA) Data Synchronization (DS) and Device Management (DM) standard protocols. As a framework, SyncML4J supports user-defined data sources. Data sources can range from simple resources, such as memos and images, to complex schema-aware data types, such as relational databases or PIM databases.

Interaction services

The client platform is built on the Eclipse Rich Client Platform (RCP) to enable the delivery of applications that provide a rich user interface across multiple platforms. The client platform provides the Workbench, Standard Widget Toolkit (SWT), JFace, Help, and Preferences Interaction services.

The Eclipse Workbench provides an integrated application desktop window to enable the ability to install, manage, and launch one or more applications within a single window. The Workbench presents each application individually in its own perspective (only one of which is visible at any given time).

The client platform supports servlets and JSPs so users can interact with local Web Applications through a Web browser. Each Web application installed onto the Workbench runs in a browser perspective. When an end user selects a Web application from the Workbench, it is launched in the local Web browser in the Web perspective.

The client platform also supports rich client applications, which interact with end users through a graphical user interface (GUI). Each rich client application installed onto the Workbench runs in an application perspective. In this case, each application must contribute its own perspective to the Workbench. In each perspective, an application provides the collection of views, layout of views, and actions appropriate for the tasks that end users will perform with the application. You use SWT or the JFace toolkit to develop the GUI for rich client applications. SWT provides a cross-platform API that tightly integrates with the native widgets of the operating system and, therefore, gives your applications a look and feel that makes them virtually indistinguishable from native applications. The JFace toolkit provides a set of components and helper utilities that simplify many of the common tasks in developing SWT user interfaces. When an end user selects a rich client application from the Workbench, it is launched in the appropriate perspective.

Platform Management services

The Platform Management service provides an enterprise with the following management capabilities:

- Device configuration: Setting device parameters for hardware or software.
- Inventory collection: Collecting hardware and software information about the device.
- Software distribution: Distributing, installing, and removing software or data files for the device.
- Initial provisioning: Providing initial access for devices to the Device Manager server and restoring the Device Manager configuration information.

There are two key WED client Platform Management services:

- *Update Manager:* The Update Manager enables end-users to directly install applications and components from standard Eclipse Update Sites onto the Workbench.
- *Enterprise Management Agent:* The Enterprise Management Agent works cooperatively with the Device Management server provided by the WebSphere Everyplace Deployment server to perform management operations. The agent and server use the SyncML/DM protocol defined by the Open Mobile Alliance to communicate management requests. An administrator can schedule management jobs for devices that include software installation, update, and configuration. When installing and updating software components, the management system determines which components are already on the device and then installs only the missing components.

From a WED server perspective, WebSphere Everyplace Device Manager (WEDM) is composed of four major components:

- *Device Manager server (DMS):* Manages and executes device management jobs in various ways in response to triggers. These triggers could include a device connecting to the server, a call to an API, or an administrator action.
- *Device plug-ins:* Provide the logic that handles device identification, communications, job processing, and other high-level management tasks.

- *Device Manager Database:* Repository for all device management information. This includes device inventory information, device management jobs, package meta data, and many other details.
- *Device Management API:* Programming interface between the Device Manager server, administration clients, or external applications (Web Services and Java).

Device agents communicate with the Device Manager server using HTTP or HTTPS. The protocol running on top of HTTP is either a proprietary protocol, which is used with Pocket PC and Palm OS devices, or a standard protocol, such as the Open Mobile Alliance, Device Agent (OMA DA) protocol, used with OSGi devices. With HTTP and HTTPS communications between devices and Device Manager, requests and responses can pass through various network elements, such as firewalls.

Note: More information about IBM WebSphere Everyplace Deployment V6.0 can be found at:

http://www.ibm.com/software/pervasive/ws_everyplace_deployment/v6/

IBM WebSphere Everyplace Access

IBM WebSphere Everyplace Access V5.1 (WEA) provides an integrated client-server environment to extend business applications to mobile users. WEA extends the value of the customer's WebSphere platform by providing mobile support for enterprise applications. WebSphere Everyplace Access can automate effective user interactions—with information, applications and business processes at any time from anywhere—accessed by thin and rich clients that are connected or occasionally connected. Use of WebSphere Everyplace Access can help enterprises transform themselves into on demand businesses by extending access to critical information and applications. For example, mobile sales force and field force automation solutions built and deployed on WebSphere Everyplace Access 5.1 can help improve productivity, increase revenue, and reduce cost of operations.

WEA provides a common back-end integration framework, common administration and management capability, and complementary functions for building, deploying, and managing mobile solutions. With WebSphere Everyplace Access 5.1, customers, independent software vendors (ISVs), and system integrators can build mobile solutions quickly and take advantage of existing skills, standards, and technologies. WebSphere Everyplace Access can support multiple applications from a single integrated framework, which simplifies skills and systems requirements. WebSphere Everyplace Access supports devices that utilize a wide range of operating systems for handheld devices, including Palm OS, Symbian, Research In Motion (RIM), Pocket PC 2002, Windows Mobile 2003, and Windows Mobile 2003 Second Edition.

Types of information:

- *Portal content adaptation:* IBM WebSphere Everyplace Access allows users to view Portal content online or selected Portal content offline. WebSphere Everyplace Access provides device enhancements for viewing the Portal online, which improves the appearance of the Portal on selected devices. WebSphere Everyplace Access also installs new page groups, in addition to the default Portal page groups, which include everything you need to get started configuring and using Offline Portal Browsing and administering your WebSphere Everyplace Access system.

  Transcoding Technology allows the reuse of information originally written in one format by converting it to a format accepted by the requesting device. For example, Transcoding Technology can convert content in HTML format to WML format for a device that supports WML only.
- *Personal Information Manager (PIM):* WebSphere Everyplace Access provides synchronization technology with Everyplace Synchronization Server (TrueSync Server), which allows users to synchronize PIM and e-mail data from back-end groupware sources such as Lotus Domino or Microsoft Exchange to their (mobile) device. This allows mobile users to have instant access to a subset of this information even in disconnected mode. The synchronization mechanism ensures that the data is kept up-to-date with the server side.
- *Relational database data:* WebSphere Everyplace Access includes both server and client components to access any JDBC-compliant relational database. DB2 Everyplace (DB2 SyncServer) provides the server components and Everyplace Client provides the DB2 Everyplace client component.
- *Messages and Notifications:* WebSphere Everyplace Access provides intelligent notification services to users across multiple delivery channels based on user preferences and subscriptions (delivery channel examples are SMS, e-mail, IBM Lotus Sametime). This functionality is referred to as Intelligent Notification Services in WEA (INS). The IBM Everyplace Client also supports the reception of Server-Initiated Action notifications. Server Initiated-Action notifications are sent by the server and cause the client to initiate commands, for example, to start the device management agent for new job retrieval or to synchronize PIM and e-mail data. Well documented APIs allow other applications to use this service.
- *Mapping and directions:* WebSphere Everyplace Access provides application providers with the ability to access location-based services from multiple vendors, while providing application developers with a common easy-to-use

Application Programming Interface (API). Adapters that interface with individual providers give provider transparency so that application developers can program to the services and not the individual providers.

- *Device Management Server (DMS):* WebSphere Everyplace Access provides the ability for an enterprise to manage a varied set of pervasive devices. Device management can be used for configuration, inventory collection, and distribution of software. The device management provided with WEA extends the OMA-DM platform-based device management in WED (for OSGi and Eclipse platforms) with additional device management agents working on operating system level (for example, device management agents for Windows Mobile or Symbian OS devices).
- *Location Awareness Server (Mapping and directions):* WebSphere Everyplace Access provides application providers with the ability to access location-based services from multiple vendors, while providing application developers with a common easy-to-use Application Programming Interface (API). Adapters that interface with individual providers give provider transparency so that application developers can program to the services and not the individual providers.

Note: More information about IBM WebSphere Everyplace Access V5.1 can be found at:

http://www.ibm.com/software/pervasive/ws_everyplace_access/

IBM WebSphere Everyplace Connection Manager

IBM WebSphere Everyplace Connection Manager V5.1 (WECM) is a distributed, scalable, multipurpose communications platform designed to optimize bandwidth, help reduce costs, and help ensure security. It creates a mobile VPN that encrypts data over vulnerable wireless LAN and wireless WAN connections. It integrates an exhaustive list of standard IP and non-IP wireless bearer networks, server hardware, device operating systems, and mobile security protocols. It:

- Provides compression and other network optimizations that increase user response time and lower network costs.
- Features seamless cross-network roaming, making it possible to dynamically switch network connections without interrupting applications.
- Encrypts data over vulnerable wireless LAN and wireless WAN connections.
- Integrates standard Internet Protocols (IP) and non-IP wireless bearer networks, server hardware, device operating systems, and mobile security protocols.

- WebSphere Everyplace Connection Manager allows service providers to offer highly encrypted, optimized, and scalable solutions to enterprise customers.
- WebSphere Everyplace Connection Manager also ships client-side components for numerous different devices. The client component provides connectivity and security services for the client.

Note: More information about IBM WebSphere Everyplace Connection Manager V5.1 can be found at:

`http://www.ibm.com/software/pervasive/ws_everyplace_connection_manager/`

IBM WebSphere Everyplace Device Manager

IBM WebSphere Everyplace Device Manager V5 (WEDM), known as Device Manager, is device management technology that helps enterprises and private networks, or providers, manage devices. Device Manager is designed to manage a wide range of devices including desktops, laptops, personal digital assistants (PDAs), mobile phones, set-top boxes, in-vehicle information systems, etc.

The DMS supports the following functions:

- *Job control:* Job control involves defining and managing the jobs that perform actions on devices. The job control function supports:
  - Targeting jobs to single devices
  - Targeting jobs to groups of devices
  - Sending *notification messages* to devices to initiate a management session
  - Queuing jobs (that is, any management activity) for devices that may not be on the network at the moment
  - Scheduling jobs to occur in the future and setting expiration times for jobs
  - Automatic re-try of jobs for certain types of failures
  - Tracking the progress of jobs and their ultimate success or failure on an individual device basis
- *Device identification:* DM can identify both OMA (SyncML) DM and non-OMA (SyncML) DM devices when they establish a connection to the DMS. Non-OMA (SyncML) DM devices are identified by platform-provided serial number when possible, and by a generated globally unique identifier otherwise. Non-OMA (SyncML) DM devices also report their device class.
- *Enrollment/initial device setup:* Enrollment occurs when a device first becomes known to the device management system. Manual enrollment takes

place when an administrator or customer service representative creates a new device entry, which may be done though one of the standard GUI interfaces or can also be done *programmatically* (for example, by a workflow process). Devices may enroll themselves automatically upon their first connection to the device management system, as long as:

  – The device provides a valid subscriber ID and credentials.
  – The system is configured to allow devices to enroll themselves.
  – At the time of first connection, a set of jobs is automatically processed for new devices. New jobs can be targeted at specific devices upon enrollment (for example, to provision *packages* of applications and settings as a part of the enrollment process).

- *Device configuration:* Device configuration allows the DMS to set various device parameters. These are specified as name/value pairs, and the standard DMS interface supports around 100 pre-defined parameters covering typical settings. Parameters that can be configured from the DMS Administration Console or through the APIs include network addresses, dialup phone number, DMS server address (URL), e-mail account information, etc.
- *Software package definitions:* Software installation and removal by the DMS is done on the basis of software package definitions.
- *Software distribution (installation or removal):* A software distribution job specifies the software package or meta-package file name that is to be installed or removed, and the class or classes of device, or the specific individual devices to which a job applies.
- *Device Inventory collection:* For asset inventory purposes, an inventory job.
- *Support for new devices:* For non-OMA (SyncML) DM devices, WEDM is extensible and can be customized through its framework and plug-in architecture. New device support can be added by creating additional device management plug-ins or by subclassing existing plug-ins. For OMA (SyncML) DM compatible devices, IBM has been working directly with manufacturers to ensure interoperability as new devices are released and as the OMA DM standard evolves.

WebSphere Everyplace Device Manager is available as a standalone product, and as a component of WebSphere Everyplace Access and WebSphere Everyplace Deployment.

Note: More information about IBM WebSphere Everyplace Device Manager V5 can be found at:

`http://www.ibm.com/software/pervasive/ws_everyplace_device_manager/`

IBM WebSphere Voice Server

IBM WebSphere Voice Server for Multiplatforms V5.1.3 (WVS) provides speech recognition and speech synthesis (Text-To-Speech or TTS) engines, voice application development tools, and telephony platform connections to develop and deploy applications for use over telephones.

WebSphere Voice Server provides the automatic speech recognition (ASR) and TTS resources that enable speech-based interaction. A speech application or VoiceXML content provides the logic and flow of dialogue. A VoiceXML browser provides access to the speech application or VoiceXML content from a Web server. The VoiceXML browser interprets the application, processing conversational interactions requiring ASR and/or TTS resources. When these elements are combined, the speech interface is perceived as a series of system utterances that require user responses. The WebSphere Voice Server allows system administrators to configure, monitor, and troubleshoot voice server resource usage in deployed speech applications.

WebSphere Voice Server for Multiplatforms includes:

- Connection to many telephony platforms, including both WebSphere Voice Response for AIX and WebSphere Voice Response for Windows, Intel Dialogic, Cisco or Siemens HiPath, and Voice Server Speech Technologies for Windows and Linux.
- Voice Toolkit for WebSphere Studio, which includes a VoiceXML editor, grammar editor, and a pronunciation builder, allows application developers to easily add voice technology to middleware applications. Tools for prototyping VoiceXML applications on a PC without a telephony server, and the necessary speech recognition and TTS engines for testing applications.

Note: More information about IBM WebSphere Voice Server for Multiplatforms can be found at:

http://www.ibm.com/software/pervasive/voice_server

IBM WebSphere Voice Application Access

IBM WebSphere Voice Application Access V5 (WVAA) extends the popular WebSphere Portal infrastructure and programming model to a voice user interface (VUI), enabling users to access a wide range of business applications and data through a standard telephone or cell phone. The WVAA voice portal platform supports VoiceXML using the WebSphere Voice Response for AIX VoiceXML browser or other interactive voice response (IVR) platforms approved by IBM and compatible with VoiceXML.

WebSphere Voice Application Access V5 allows users to take advantage of the personalization features of WebSphere Portal to tailor their individual voice portals to fit their needs.

For the administrator, it provides a consistent framework for administering users and extending the same portal security features for authentication and authorization across multiple channels.

For the developer, WebSphere Voice Application Access V5 leverages the same Eclipse-based programming model as WebSphere Portal to build applications using VoiceXML technology. Developers can reuse existing code, business logic, and infrastructure, and enhance the development process to build VoiceXML applications.

Note: More information about IBM WebSphere Voice Application Access can be found at:

`http://www.ibm.com/software/pervasive/ws_vaa/`

IBM WebSphere Voice Response for AIX

IBM WebSphere Voice Response for AIX is capable of supporting simple to complex applications and can scale to thousands of lines in a networked configuration. Applications can be developed using the native development environment or using standards-based development tools for Java or VoiceXML. WebSphere Voice Response Server is used to link the telephony users into the corporate Web environment.

- WebSphere Voice Response provides an Interactive Voice Response (IVR) system for medium and large enterprises, telcos, and service providers.
- WebSphere Voice Server software resources for developing and deploying speech solutions on telephony platforms.
- WebSphere Voice Application Access voice portal enablement for telephony platforms.
- WebSphere Voice Response supports speech recognition technologies that can replace traditional telephone keypad input with more natural voice responses.
- WebSphere Voice Response is highly scalable to support thousands of Public Switched Telephone Network (PSTN) connections.
- WebSphere Voice Response combined with WebSphere Studio enables Java and VoiceXML service creation environments for developing Web and telephone self-service applications that are easy to integrate.

- WebSphere Voice Response for AIX complements WebSphere Voice Server V5.1, making it possible to create integrated Web and telephone self-service access to your business data and processes.
- WebSphere Voice Response includes VoiceXML and Call Control XML (CCMXL) industry-standard programming environments that enable telephony channel handling and voice call processing that can be integrated with WebSphere Portal Server and WebSphere Voice Application Access. This produces a Business Portal Multichannel access point to deliver voice-enabled e-business solutions. WebSphere Voice Server can be included for speech-enabled applications such as speech recognition or text-to-speech. If required, WebSphere Voice Response and WebSphere Voice Server can be used together for non-IBM application server solutions.
- Contact/call center solutions: WebSphere Voice Response is the self-service channel in many contact/call center solutions. It can front-end the call center in front of the PBX or ACD switch, or be behind the ACD/switch and also support the call center agents. With new VoIP SIP support, it can also be used in IP contact/call center solutions alongside the main Computer Telephony Integration (CTI) suppliers in the industry. Self-service applications can support conversational speech, enabling more natural interaction with customers in a user-friendly manner.

Note: More information about IBM WebSphere Voice Response for AIX can be found at:

`http://www.ibm.com/software/pervasive/voice_response_aix/`

IBM WebSphere Application Server

IBM WebSphere Application Server V6.0 provides the foundation of the WebSphere software platform. WebSphere Application Server V6.0 is the IBMs Java-based application platform, integrating enterprise data and transactions for the dynamic e-business world. Each configuration available delivers a rich application deployment environment with application services that provide enhanced capabilities for transaction management, as well as the security, performance, availability, connectivity, and scalability expected from the WebSphere family of products.

WebSphere Application Server V6.0 is the premier Java 2 Enterprise Edition (J2EE) and Web services technology-based application platform, delivering a high-performance and extremely scalable transaction engine for dynamic e-business applications. It provides:

- J2EE 1.4 programming model

  - Supports the J2EE 1.4 programming model and extensions including Servlets, JSPs, EJBs, and Web services, as well as additional programming model enhancements, to provide a secure foundation for a services oriented architecture.
  - Dramatically simplifies the task of connecting applications to an ESB with JMS 1.1.
  - Enables more demanding business applications by leveraging SDK 1.4.2
  - Easily connects to multiple back-end data sources through a standard interface with Service Data Objects (SDO).
  - Build dynamic Web user interfaces with drag-and-drop development using standards-based JavaServer Faces (JSF).
- Ease of use
  - Provides WebSphere Rapid Deployment for reducing the complexity of developing and deploying a J2EE application
  - Supports mixed application server support (V5, V5.1, and V6) for more flexible migration to newest versions
- Tight security
  - Integrates closer with Tivoli offerings, including embedded Tivoli Access Manager, for centralized security management among J2EE and Web Resources
  - Supports Web Services Security open security model for easy interoperability with third-party, customer-created, or legacy security solutions
- Platform and performance
  - Outstanding performance for better business results
  - Broad platform support, allowing increased flexibility in deployment choice and a unified administration across disparate systems, including Linux on iSeries™, pSeries®, and zSeries

Note: More information about IBM WebSphere Application Server V6.0 can be found at:

`http://www.ibm.com/software/webservers/appserv/was`

IBM WebSphere Portal

IBM WebSphere Portal for Multiplatforms V5.1 offers a single point of personalized interaction with applications, content, processes, and people. To deliver a unified user experience, WebSphere Portal brings together a range of leading-edge technologies designed to give you a flexible, open, extensible framework to build successful business-to-employee (B2E), business-to-business (B2B), and business-to-consumer (B2C) portals.

- Through business process integration, WebSphere Portal combines people and applications at a process level. The result is that people are more productive and processes are executed faster. Portal's navigation paradigm is not only role-based, but also includes workflow orchestration that presents users with the tasks they need to complete and all information and applications needed to complete the task or decision quickly.
- Create multiple portal sites on one instance of WebSphere Portal. Each site has its own URL, look and feel, pages, users and groups, and search index. All sites can share the same software and hardware, which lowers capital, maintenance, and administration costs while expanding the business value of the portal to new communities.
- An integrated version of IBM Workplace Web Content Management™ is included with WebSphere Portal and keeps the portal up-to-date with accurate content created by portal users (usage restrictions apply).
- The user interface for Portal Document Manager allows sharing, viewing, and organizing files of all types ranging from documents to spreadsheets within the Portal community. It also offers category subscription services, simple approval processes for file contribution, versioning so that users can track the evolution of a piece of content, and access control for managing viewing privileges of different content items.
- DB2 Content Manager runtime repository manages all forms of content (Web, e-mail, documents, digitized paper documents, images, audio/video, text messages) consistently.
- Portal search technology allows a user to dynamically search across all portal content while adhering to access control limits.

Additionally, WebSphere Portal Enable provides the following functions that help to improve employee productivity and customer loyalty:

- Portal Application Integrator, which allows business users to quickly create portlets for interacting with relational databases, Domino databases, and enterprise applications from Oracle, SAP, Siebel, and PeopleSoft.
- Click-to-action (C2A) technology for portlet-to-portlet communication and action, ensuring accuracy of information passed and delivering it on demand.
- Productivity Components, which allow users to view, create, convert, and edit basic documents, spreadsheets, and presentation files from the portal interface. Therefore, they can execute ad-hoc business process from the same place they access their applications, search for information, and

collaborate with other employees and partners. The productivity components are integrated with the document management feature so files can be indexed, categorized, and searched by other portal users.

- Integration services that give access to enterprise data, applications, newsfeeds, and Web services.
- Publish local portlets as remote Web services or subscribe to Web services to make them available to portal users via portlets.
- Presentation services that allow for the customization of the computing desktop to match individual work patterns and roles.
- Powerful personalization technology so portal users get a unique experience based on their role and business rules.
- WebSphere Translation Server functionality, which helps you to translate the contents of portlets from English to French, Italian, German, Spanish, Portuguese, Taiwanese, Japanese, Simplified Chinese, and Traditional Chinese. Or you can translate your portlet content from those languages to English.
- Access Lotus and Microsoft Office applications via portlets.
- Use a flexible architecture that enables integration with your current directory, database, and security infrastructure.

Note: More information about IBM WebSphere Portal Enable for Multiplatforms V5.1 can be found at:

`http://www.ibm.com/software/genservers/portal/enable`

IBM DB2 Universal Database

IBM DB2 Universal Database V8.2, Enterprise Server Edition is a full-featured, robust, scalable, and easy-to-use relational database. DB2 UDB provides the foundation of information about demand on Linux, UNIX®, and Windows platforms.

Innovative manageability DB2 Version 8.2 provides automation capabilities, including self-configuring, self-optimizing, and self-managing capabilities.

- New levels of integrated information across the entire enterprise leverage federated Web Services and XML to help solve critical business problems. New federated capabilities enable customers to integrate information as Web Services. XML enhancements make it easier for programmers to integrate DB2 and XML information.

- Robust e-business foundation: Performance, scalability, and availability enhancements continue with cross-workload and cross-platform leadership, improving overall application performance and making information highly available.

Note: More information about IBM DB2 Universal Database V8.2, Enterprise Server Edition can be found at:

`http://www.ibm.com/software/data/db2/udb/edition-ese.html`

IBM WebSphere Everyplace Client

IBM WebSphere Everyplace Client is a client-side application that enables you to synchronize data between your device and database, gain access to offline home pages, perform device management, and receive software updates on your device.

Everyplace Client supports the following list of features:

- E-mail and Personal Information Management (PIM)
- Database synchronization
- Offline browsing and forms
- Device Manager Server (DMS) access
- Everyplace Client user interface
- Sametime Connect access
- Server initiated actions
- Mobility Client
- Workplace Client Technology, Micro Edition (WCTME)
  - Extension Services for WebSphere Everyplace (ESWE)
  - Mobile Information Device Profile (MIDP)

IBM Everyplace Client is included with WebSphere Everyplace Access.

Note: More information about IBM Everyplace Client can be found at:

IBM Lotus Domino

IBM Lotus Domino server provides enterprise-grade collaboration capabilities that can be deployed as a core e-mail and enterprise scheduling infrastructure (IBM Lotus Domino Messaging Server), as a custom application platform (IBM Lotus Domino Utility Server), or both (IBM Lotus Domino Enterprise Server).

An integral part of the IBM Workplace family, Lotus Domino server and its client software options deliver a reliable, security-rich messaging and collaboration environment that helps companies enhance the productivity of people, streamline business processes, and improve overall business responsiveness.

Note: More information about IBM Lotus Domino can be found at:

http://www.ibm.com/software/sw-lotus/products/product4.nsf/wdocs/dominohomepage

IBM Lotus Sametime

IBM Lotus Sametime is IBM's market-leading product and platform for real-time collaboration. It is based on three on demand concepts:

- *Presence awareness:* See, in advance, whether a person or application is available to collaborate and share information.
- *Instant messaging:* Be able to converse virtually through the exchange of text-based, audio-based, and/or video-based information in real time.
- *Web conferencing:* Share information, an application, or an entire desktop or engage in team white boarding.

Though basic in nature, these capabilities present customers with virtually unlimited possibilities. In fact, over fifteen million people worldwide use Lotus Sametime capabilities every day to gain instant access to people and information, bring together geographically dispersed team members, and improve individual and team productivity.

Lotus Sametime also takes advantage of new audio integration capabilities from leading teleconferencing and telecommunications providers to offer a single, unified interface to both audio and Web conferencing, as well as click-to-call functionality directly from the Connect Client.

Note: More information about IBM Lotus Sametime can be found at:

http://www.ibm.com/software/sw-lotus/products/product3.nsf/wdocs/homepage

IBM Workplace Collaboration Services

IBM Workplace Collaboration Services is a single product that provides a full range of integrated ready-to-use communication and collaboration tools that enable people to do their jobs more effectively—anytime, anywhere. With the flexibility to deploy any mix of capabilities, it provides a ready-made foundation to build customized role-based Workplace environments that provide the tools and information people need to do their work more efficiently.

Available as a fully integrated product or with the option to purchase the capabilities separately such as Workplace Messaging® (see Licensing options below), Workplace Collaboration Services is built on a componentized architecture that is adaptable to enable people and teams to react quickly to changing business needs. IBM Workplace products provide the front-end to the IBM service-oriented architecture (SOA) strategy.

The following services are provided by Workplace Collaboration Services:

- Team Collaboration Services
- Document Services
- Messaging Services
- Web Content Management Services
- Learning Services
- Workplace Managed Client

Team Collaboration Services

The IBM Workplace Team Collaboration™ Services component provides users with the capability to participate in online meetings, create libraries, and interact with team members through online chats, threaded discussion forums, and document sharing. It includes the following features:

- *Applications* provide users with access to Workplace applications, HTML-enabled Domino applications, and custom applications. Workplace applications include: *Team Spaces*, where members can participate in discussions/chats, share documents and a team calendar, and search for information; and *Documents* for managing online document libraries. Users can create and maintain a list of favorite document libraries.
- *Web Conferences* are online meetings in which moderators make presentations to conference participants.
- *Templates* provide tools for creating, customizing, and managing application templates, which are used in applications.

Document Services

The IBM Workplace Document Services component provides systematic, controlled access to critical documents and provides a fundamental document-management capability that is standards-based and has integrated collaborative capabilities. Workplace Documents supports both the IBM Workplace browser and the IBM Workplace rich client.

The IBM Workplace Document Services component provides access to the following:

- *Document library capabilities* provide document check-in and check-out, document locking, and version control.
- *Structured access* provides an easy method for setting up library access so that information needed organization-wide can be easily viewed, but selective information can be viewed only by a limited audience.
- *Document editors* provide the power to modify popular document types even when native editors are unavailable.
- *Document author/owner/editor awareness* through integrated instant messaging and chat capabilities.
- *Security* lets users store documents outside the file system to increase protection from viruses and other risks.

The IBM Workplace Messaging Services component for the rich client provides the following:

- *Offline support* provides a secure method for users to create, import, edit, and save documents, presentations, and spreadsheets by supporting offline use and synchronization between local and server stores.
- *Productivity tools* provide the power to modify popular document types even when native editors are unavailable.

Messaging Services

IBM Workplace Messaging Services component is a cost-effective, standards-based messaging product that is security-rich, scalable, and easily deployed. It integrates with an organization's existing corporate infrastructure and uses the organization's LDAP directory to automatically create, delete, and authenticate user accounts; resolve addresses; and route mail. Workplace Messaging supports both the IBM Workplace browser and the IBM Workplace rich client.

The IBM Workplace Messaging Services component provides access to the following:

- *Mail* lets users send and receive e-mail messages.
- *Calendar and Scheduling* lets users maintain and manage calendar events and schedule meetings.
- *Personal Address Book* lets users maintain and manage contact information for people and for group mailing lists.

The IBM Workplace Messaging Services component for the rich client provides:

- *Offline support* that allows users to read, edit, and create mail while disconnected from the network.
- *Integrated Instant Messaging* and chat, including the ability to save chats. (To make this available to users, you must have a license for IBM Workplace Collaboration Services or a license for IBM Workplace Team Collaboration, and you must configure instant messaging.)

Web Content Management Services

The IBM Workplace Web Content Management Services component delivers powerful end-to-end Web content management through multiple Internet, intranet, extranet, and portal sites.

IBM Workplace Web Content Management is not installed as part of IBM Workplace Collaboration Services (installed separately):

- *Content authoring* is template-based, with a WYSIWYG rich text editor providing a guided process that does not require technical skills.
- *Versioning and rollback* provides a method for creating multiple content versions that can be used at different times or restored to previous versions, as needed.
- *Automatic workflow processing* ensures that the right people approve Web content before it is published and assures accuracy and relevancy of content.
- *Integration of information from various sources* allows reuse of information from back-end systems, improving transactional performance.
- *Personalized delivery* lets authors create content once and reuse it in different sites for users with different roles or preferences.
- *Multiple database support* allows use of IBM Lotus Domino, IBM DB2, Oracle, or IBM DB2 Content Manager as repositories.

Learning Services

IBM Workplace Learning Services component (Collaborative Learning) provides access to a scalable, flexible product for managing classroom-based and online learning activities, resources, curricula, and courseware catalogs.

IBM Workplace Collaborative Learning™ provides access to these features:

- *The learning student experience* provides an easy-to-use interface where students access courses. Students can search for courses and organize them in personalized folders, as well as preview, enroll in, and participate in courses online. Information about courses is stored on the Learning Server, while the courses themselves are presented on the Delivery Server.

- *The learning management and delivery system components* provide a Web-based administrative interface that course developers and instructors access to manage resources, learning programs, and skills development.
- *The Authoring Tool* is used by course developers on their own workstations to create course structure and content, assemble course packages, and import courses to Learning servers.

Workplace Managed Client

The IBM Workplace Managed Client provides rich client capabilities to the IBM Workplace Collaboration Services product. Users can access collaboration capabilities from their desktop, rather than from a browser.

The IBM Workplace Managed Client is a separate component, which provides users with access to offline use and replication, as well as to these features:

- *IBM Activity Explorer* is used to track and manage activities related to a project or process. Users can capture and manage real-time communications and leverage the collaboration features of shared workspaces.
- The *IBM data access tool* is used to create relational database applications. Users can create forms, grids, and reports in order to add, edit, delete, and view summaries of database records stored in IBM Cloudscape databases.
- On *Embedded browser* is what users use to open and navigate Web pages directly from within the rich client.

Note: More information about IBM Workplace Collaboration Services can be found at:

http://www.ibm.com/software/workplace/products/product5.nsf/wdocs/workplacehome

B

Description of IT drivers and capabilities

In Chapter 4, “Business and IT drivers” on page 83, we provided a summary of the IT drivers and capabilities to be used as selection criteria for the appropriate Application pattern or patterns. This appendix provides a more detailed description of both the IT drivers and capabilities.

The appendix is organized into the following sections:

- IT drivers description
- Capabilities description

IT drivers description

This section provides a detailed description of the IT drivers that were summarized in Table 4-1 on page 89 in Chapter 4, “Business and IT drivers” on page 83.

IT01: Multi device access to Web applications

Extending access to corporate Web applications by tailoring the Web content for a class of devices.

For example, extending access to corporate information such as an online telephone book, and mobile access to a corporate intranet. In this case the range of supported devices is limited. Typically the mobile view of the application will be created during the development process.

IT02: Write once, render many

Device-independent authoring to support to concept of *write once, render many* to a broad range of users and mobile devices. When operating in the public domain such as the Internet, the devices types change frequently and can be very unpredictable. There is a need to provide a dynamic mechanism to support new devices when they emerge in the market without changing the business applications.

Typical examples include mobile information services such as mobile news, commercials, mapping and routing services, customer services like mobile online banking, etc.

IT03: Common programming model

Use of an existing programming model to leverage existing assets, programming skills, and techniques. Provide the same techniques and tools on server and mobile domain. Typical examples include model-view-controller (MVC) design pattern, J2EE, and .NET frameworks.

IT04: Single sign-on

Provide secure and seamless access to applications and information. Multi sign-ons to applications are cumbersome, especially on mobile devices with a limited user interface. Single sign-on will optimize the logon processes to back-end systems.

Typical example are a personal workplaces (Portal) with access to collaboration and business applications.

IT05: Role based personalized content, customized look and feel

Access to information and applications based on users roles. Ability to customize the look and feel following corporate rules and design guidelines as well as user preferences. IT drivers 1–4 can also be included to add content adaptation and single sign-on with this IT driver.

For example, a corporate portal with access to a Human Resource Manager (role) with a personalized workplace that provides fast access to human resource applications.

Additional IT drivers: IT01–04

IT06: Location Aware Services

Provide location information to applications and users using Location Aware Services. Provide a consistent and service provider independent programming model.

Typical examples include tailored information based on location like points-of-interest, location of closest ATM machine, or pre-filled formulas like start point on routing application or train schedules.

Additional IT driver: IT03

IT07: Voice access

Extend access to information and applications by voice access. Leverage existing artifacts and skills by using a common programming model.

Typical examples include customer services such as voice access to time schedules for trains or flights, banking account information, and self-service reservations systems.

Additional IT driver: IT03

IT08: Simple push notification

Provide a simple push notification capability to send simple messages.

Typical examples include SMS broadcasts from service providers and SMS commercials.

Additional IT driver: IT03

IT09: Alert user/application of defined state changes

Provide subscribe and publish services to users and applications. Enable subscription, trigger, and notification functionality, allowing you to monitor state changes of different resources (for example, database, news feeder, e-mail inbox, etc.). Provide a standard programming interface to build and extend notification services.

Typical examples include notification services such as news subscriptions, monitoring stock quotes, and account balance.

Additional IT driver: IT03

IT10: Actionable alerts

Provide the ability to send actionable alerts to a mobile device in order to start actions transparently on the device (for example, start synchronization process).

Typical examples include automatic PIM/e-mail synchronization (push mail), and automatic installation of security updates on the device (push from server).

IT11: Distribute device configuration

Allows you to distribute device configurations such as agent and security settings to mobile devices. Typical examples are provisioning of initial device settings and boot-strapping configuration.

Additional IT drivers: IT03, IT10

IT12: Rollback and recovery mechanism during task execution

Ability to roll back or recover a running task in case of errors. Typical examples include recovery of sychronization status after loss of connectivity. Roll back in case of issues during a software installation such as running out of memory.

IT13: Inventory

Collect and maintain inventory information off mobile devices. This information allows for a complete view of the device platform.

Typical examples include support and security organizations that need to maintain control of the installed components on the device.

IT14: Remote maintenance of middleware services

Provide distribution, update, and removal of middleware services on mobile devices. Additional IT drivers such as inventory, rollback, and actionable may also apply to this driver.

For example, the installation and maintenance of an OSGI framework on mobile devices.

Additional IT drivers: IT03, IT10, IT12, IT13

IT15: Remote maintenance of applications

Provide distribution, update and removal of application services on mobile devices. Additional IT drivers such as inventory, rollback, and actionable may also apply to this driver.

For example, installation and maintenance of an application on mobile devices.

Additional IT drivers: IT03, IT10, IT12, IT13

IT16: Cache Web application for offline viewing

Provide the ability to cache simple Web data on mobile devices for offline viewing. Content adaptation and roles-based information access are complementary IT drivers.

Typical examples include simple news and weather Web pages.

Additional IT drivers: IT01, IT02, IT05

IT17: Offline forms

Enable the use of forms in disconnected mode. Provide the ability to collect data in forms for later update with the back-end. Content adaptation, role based access as well as simple offline Web caching may also apply.

For example, a claim form of an insurance adjuster on a mobile device.

Additional IT drivers: IT01, IT02, IT05, IT16

IT18: Disconnectable application without GUI

Run a headless (no GUI) application on a mobile device. Disconnectable application that does not have a Graphical User Interface (GUI).

Typical examples include applications running on devices such as RFID reader, sensors in cars and vending machines, laptops, and handheld computers. Typical types of applications are device agents and data collection applications.

Additional IT drivers: IT03, IT11–IT15

IT19: Disconnectable application with Web User Interface

Run disconnectable applications that have a Web User Interface on the local device. The client platform will provide a local Web container. This will introduce the traditionally server based programming model to mobile platforms. Device management is recommended as a complementary IT driver.

Additional IT drivers: IT03, IT11–IT15, IT18

IT20: Disconnectable relational database sync services

Work with a local database on the device and synchronize data to the back-end using database sync technology. Using standard SQL API will leverage the existing database programming model.

Additional IT driver: IT03

IT21: Disconnectable transactional messaging services

Extend the messaging paradigm by enabling transactional services such as transaction security, message queueing, and assured delivery to the mobile domain.

IT22: Client Web services

Provide and consume Web services on the mobile device.

IT23: Standalone rich client User Interface

Provide a rich client user interface using standard frameworks such as JFace and SWT libraries to standalone applications. This provides a common programming interface. Device management and Web service capabilities are recommended supporting IT drivers.

Additional IT driver: IT03, IT11–IT15, IT20, IT22

IT24: Disconnectable rich client User Interface

Provide mobile applications with a rich client user interface using standard local frameworks such as JFace and SWT libraries. This provides a common programming interface. Device management, messaging, and Web service capabilities are recommended supporting IT drivers.

Additional IT driver: IT23

T25: Extend disconnectable Web apps for text and voice

Extend disconnectable applications with a Web user interface that can use both text and voice. Provide a multimodal user interface that supports data and voice interaction by using XML standards such as XHTML and VoiceXML.

Device management and the disconnectable applications are recommended supporting IT drivers.

Additional IT drivers: IT03, IT11–IT15, IT20–22

IT26: Provide instant messaging

Enable people awareness and instant messaging on mobile devices. Additionally, leverage notification services. Device management is a recommended IT driver.

Additional IT drivers: IT08, IT09, IT11–IT14

IT27: Provide offline PIM and e-mail access

Provide synchronization of PIM/EMail to the mobile device. Optionally enable push e-mail services. Device management drivers are a recommended set of related IT drivers.

Additional IT drivers: IT09, IT11–IT14

IT28: Extend collaboration services

Provide a collaboration workplace on the mobile device providing PIM/EMail access, instant messaging, people awareness, and the ability to view or edit office documents.

Additional IT driver: IT26, IT27

IT29: Optimized remote connection

Provide secure access to the corporate network. Optimize and manage mobile connection (for example, roaming, session hold, compression, etc.).

Additional IT driver: IT11

IT30: Support of existing devices

In many cases, the customer may already have a device and want to leverage this to extend their business operations to a wide range of devices (laptop, PDA, mobile phone, embedded). To a certain degree, the device constraint will greatly define what capabilities are possible. Capturing the device and network requirements (in use, or possible) is critical.

IT31: Standalone disconnectable relational database

Work with a local database without synchronization.

IT32: Disconnectable application with performance focus

An application may require that data be captured in a disconnected mode and synchronize data at a chosen time for response time and scalability reasons (for example, filtering of RFID data).

Capabilities description

This section provides a detailed description of the capabilities that were summarized in Table 4-2 on page 94 in Chapter 4, “Business and IT drivers” on page 83.

C01: Multi device access - Simple browsing

C01: Multi device access - Simple browsing	
Capability description	► Multi device browser access. ► Device-independent authoring (write once, render many). ► Dynamic content adaptation by.
Client model	Thin Client (TC).

C01: Multi device access - Simple browsing	
IT drivers that map to capability	Provide device independent browser access to business application and information.
Example	Mobile information services such as online banking, travel- and routing information, etc.

C02: Multi device access - Role-based browsing

C02: Multi device access - Role-based browsing	
Capability description	► Role-based multi device browser access with personalized and customized content. ► Device independent authoring (write once, render many). ► Highly dynamic content adaptation.
Client model	Thin Client (TC).
IT drivers that map to capability	Provide device independent, role based browser access to business application and information.
Example	Mobile portal access to tailored information such as online banking, corporate workplace, etc.

C03: Location Aware Service integration

C03: Location Aware Service integration	
Capability description	Integrate location information.
Client model	Thin Client (TC).
IT drivers that map to capability	Provide location information to applications and users using Location Aware Services.
Example	Map of points-of-interest or closest ATM.

C04: Voice user interface

C04: Voice user interface	
Capability description	Native voice user interface ► Telephone ► Voice-over-IP (VoIP) client
Client model	Voice Enabled Client (VEC).
IT drivers that map to capability	Provide voice access to applications and information.
Example	Customer self-service such as travel and reservation service.

C05: Online multi-modal access

C05: Online multi modal access	
Capability description	Multi modal access to applications and information ► Voice access ► Data access (browsing)
Client model	Thin Client (TC), Build-in Client (BIC).
IT drivers that map to capability	Provide multi modal access to applications and information.

C06: Receive event notifications

C06: Receive event notifications	
Capability description	Receive event notifications.
Client model	Distributed Collaboration Client (DCC).
IT drivers that map to capability	Provide event notification capability to send messages over different channels.
Example	Receive event notification over different channels such as SMS, instant messaging, voice, e-mail, etc.

C07: Subscription-based event notification

C07: Subscription based event notification	
Capability description	► Subscribe to event notifications. ► Receive notification (C06).
Client model	Thin Client (TC).
IT drivers that map to capability	Provide subscribe and publish services to users and applications.
Example	Event subscription for sport events, news, stock quotes, etc.

C08: Receive and process actionable alerts

C08: Receive event notifications	
Capability description	Receive and process actionable alerts.
Client model	Distributed Application Client (DAC).
IT drivers that map to capability	Provide "wake-up" and automatic processing on mobile devices.
Example	Automatic PIM/e-mail synchronization (push mail) and device update.

C09: Remote management of device configurations

C09: Remote management of device configurations	
Capability description	Remote management of device configurations using OMADM standards.
Client model	Build-in Client (BIC).
IT drivers that map to capability	Remote management of client configurations.
Example	Provisioning of initial device settings and boot-strapping configuration.

C10: Remote management of middleware

C10: Remote management of middleware	
Capability description	► Remote management of middleware on the mobile device. ► Use OMADM standards.
Client model	Build-in Client (BIC).
IT drivers that map to capability	Remote management of mobile middleware.
Example	Remote install, update, and remove of middleware on mobile device.

C11: Remote management of applications

C11: Remote management of middleware	
Capability description	Remote management of applications on the mobile device.
Client model	Distributed Application Client (DAC).
IT drivers that map to capability	Remote management of mobile applications.
Example	Remotely install, update, and remove applications on the mobile platform.

C12: Disconnectable Web content presentation

C12: Disconnectable Web content presentation	
Capability description	View/read Web content while disconnected (offline browsing).
Client model	Distributed Presentation Client (DPC).
IT drivers that map to capability	Allow caching simple Web data on mobile devices for offline viewing without changing the Web application.
Example	Offline access to simple Web page like news, weather, travel information, etc.

C13: Disconnectable forms processing

C13: Disconnectable forms processing	
Capability description	► Work with forms while disconnected. ► Subsequent submission of forms when connected. ► Limited field validation but no interleaved business logic.
Client model	Distributed Rich Client (DRC).
IT drivers that map to capability	Work with forms in connected and disconnected mode.
Example	Data collection claim form of an insurance adjuster.

C14: Disconnectable headless application

C14: Disconnectable headless application	
Capability description	Disconnectable rich client application without a user interface (user transparent).
Client model	Distributed Application Client (DAC).
IT drivers that map to capability	Ability to run headless disconnectable applications on mobile devices.
Example	Client software on a RFID reader, sensors, device management.

C15: Disconnectable application with local Web user interface

C15: Disconnectable application with local user interface	
Capability description	► Rich function disconnectable application on the mobile device. ► Provides a local Web container. ► Use of a local Web UI or remote browser to access the application.
Client model	Distributed Rich Client (DRC).
IT drivers that map to capability	► Ability to access the disconnectable application by a browser. ► Unified browser access to disconnectable applications.

C15: Disconnectable application with local user interface	
Example	Mobile Workplace with standard browser interface.

C16: Disconnectable application using relational sync

C16: Disconnectable Applications using relational sync	
Capability description	► Rich function disconnectable application on the mobile device. ► Relational data storage on the mobile device. ► Provide relational synchronization mechanism to keep data current between two or more databases.
Client model	Distributed Rich Client (DRC).
IT drivers that map to capability	Provide relational database services and sychronization.
Example	Mobile sales and field force automation using mobile database such as catalog or customer information in disconnected mode and synchronize data between back-end database when connected.

C17: Disconnectable application using transactional messaging

C17: Disconnectable Applications using transactional messaging	
Capability description	► Rich function disconnectable application on the mobile device. ► Transactional messaging principals on the mobile device—execute distributed transactions between nodes. ► Assured delivery services.
Client model	Distributed Rich Client (DRC).
IT drivers that map to capability	Provide transactional messaging services on the mobile device for disconnectable applications.
Example	Process transactional sensitive data on a mobile device.

C18: Disconnectable application using Web services

C18: Disconnectable Applications using Web services	
Capability description	► Rich function disconnectable application on the mobile device. ► Consume and provide Web services.
Client model	Distributed Rich Client (DRC).
IT drivers that map to capability	Provide Web services on the mobile device for disconnectable applications.

C19: Disconnectable application using local rich user interface

C19: Disconnectable Applications using rich user interface	
Capability description	► Rich function disconnectable application on the mobile device. ► Rich user interface to disconnectable application using client framework and libraries.
Client model	Distributed Rich Client (DRC).
IT drivers that map to capability	Provide a rich user interface to disconnectable applications.
Example	Java user interface to a rich disconnectable application.

C20: Disconnectable application using Multimodal user interface

C20: Disconnectable Applications using multimodal user interface	
Capability description	► Rich function disconnectable application on the mobile device. ► Multimodal interface to disconnectable application.
Client model	► Distributed Rich Client (DRC).
IT drivers that map to capability	► Provide a multimodal user interface to disconnected applications that supports data and voice interaction by using XML standards such as XHTML and VoiceXML.

C20: Disconnectable Applications using multimodal user interface	
Example	Navigation system with voice and touchscreen controls.

C21: Instant messaging and people awareness

C21: Instant messaging and people awareness	
Capability description	► Instant messaging services. ► People awareness.
Client model	Distributed Collaboration Client (DCC).
IT drivers that map to capability	Enable people awareness and instant messaging between workers across device and network boundaries.
Example	Collaboration services.

C22: Disconnectable PIM and e-mail access

C22: Disconnectable PIM and e-mail access	
Capability description	► PIM/e-mail synchronization. ► Keep PIM/e-mail data current between different PIM/e-mail nodes.
Client model	Distributed Collaboration Client (DCC).
IT drivers that map to capability	Provide disconnectable access to PIM/e-mail.
Example	Collaboration services.

C23: Distributed collaboration services

C23: Distributed collaboration services	
Capability description	Interaction/collaboration between people is the central point (messaging, team meetings, productivity applications, e-meetings)
Client model	Distributed Collaboration Client (DCC)

C23: Distributed collaboration services	
IT drivers that map to capability	
Example	Collaboration services.

C24: Connectivity services

C24: Connectivity services	
Capability description	Connection optimization and management ► Security ► Roaming ► Compression ► Session management
Client model	► Distributed Collaboration Client (DCC) ► Build-in Client
IT drivers that map to capability	Provide secure connection to the corporate network.
Example	Collaboration services.

C

Rational Unified Process (RUP) overview

The Rational Unified Process is a commercial software development process framework. The RUP is composed of:

- Best practices: The RUP includes a library of best practices for software engineering, covering everything from project management to detailed test guidance.
- Process delivery tools: The RUP is delivered using Web technology, allowing it be integrated with other software development tools and making it easily accessible to developers.
- Configuration tools: The RUP is made up of components and plug-ins that can be selected and configured to meet the needs of different kinds of projects.
- Process authoring tools: An organization can extend or modify the RUP by creating its own plug-ins using the Rational Process Workbench® product.
- Community/marketplace: The Rational Developer Network® (RDN™) provides a place for process engineers in the software development community to share their process extensions.

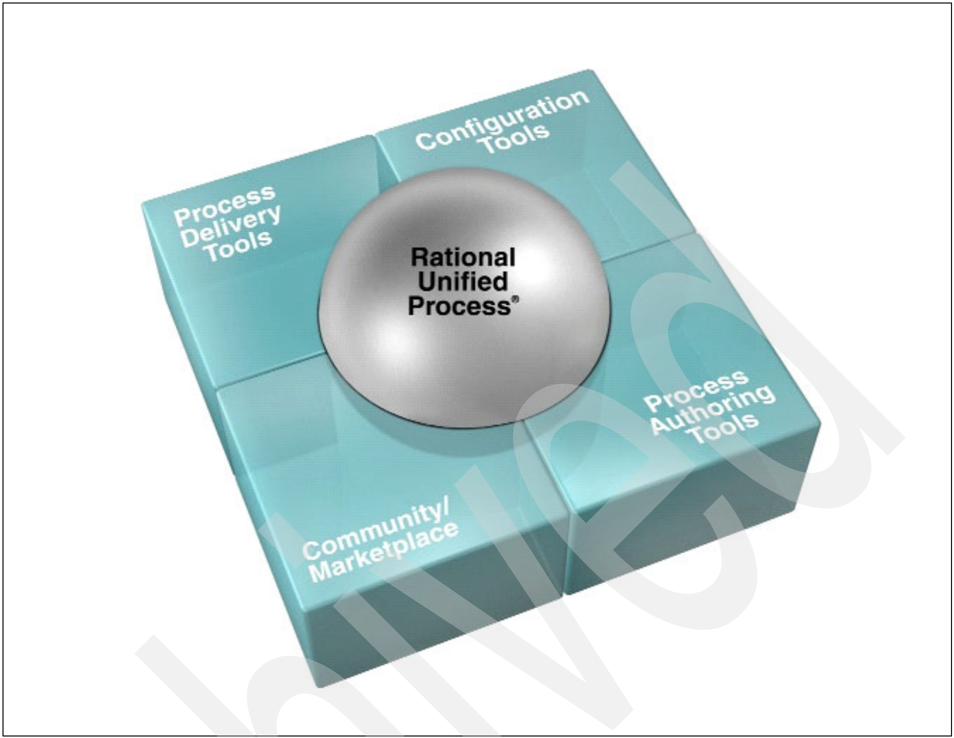


Figure C-1 Rational Unified Process

RUP is a process framework consisting of process delivery tools, configuration tools, process authoring tools, and a community/marketplace of process plug-ins.

Software development best practices

The RUP is a software development process framework that provides a disciplined approach to software development. Its goal is to ensure the production of high-quality software that meets the needs of its end users within a predictable schedule and budget.

Central to meeting this goal is promoting commercially proven best practices. These best practices include:

- ► Develop iteratively.
- ► Manage requirements.
- ► Use component-based architectures.
- ► Visually model software.

- Continuously verify software quality.
- Control changes to software.

Develop iteratively

System functionality should be delivered in a successive series of releases, addressing critical risks and getting feedback in the early releases.

Manage requirements

Requirements management is a systematic approach to eliciting, organizing, communicating, and managing the changing requirements of a software-intensive system or application.

Effective requirements management enables better control over customer satisfaction, budget, and schedule.

Use component-based architecture

The Rational Unified Process provides a methodical, systematic way to design, develop, and validate an architecture. Architectures based on components are more flexible, reusable, and understandable than architectures based on other approaches.

Visually model software

Models are simplifications of reality; they help us to understand and shape both a problem and its solution and to comprehend large, complex systems that we could not otherwise understand as a whole. RUP uses the Unified Modeling Language (UML), a standard graphical language for visualizing, specifying, constructing, and documenting software-intensive systems.

Continuously verify quality

Quality is the responsibility of every member of the development organization; it is not added by specialized staff at the end of the project. Quality is managed throughout the life cycle by implementing and assessing both process quality and product quality.

The architecture of the RUP

Figure C-2 on page 486 shows the overall architecture of the Rational Unified Process. The process has two dimensions:

- The horizontal dimension represents time and shows the phase and iteration milestones that occur over the life of the project.

- The vertical dimension represents content in logical groupings called *disciplines*.

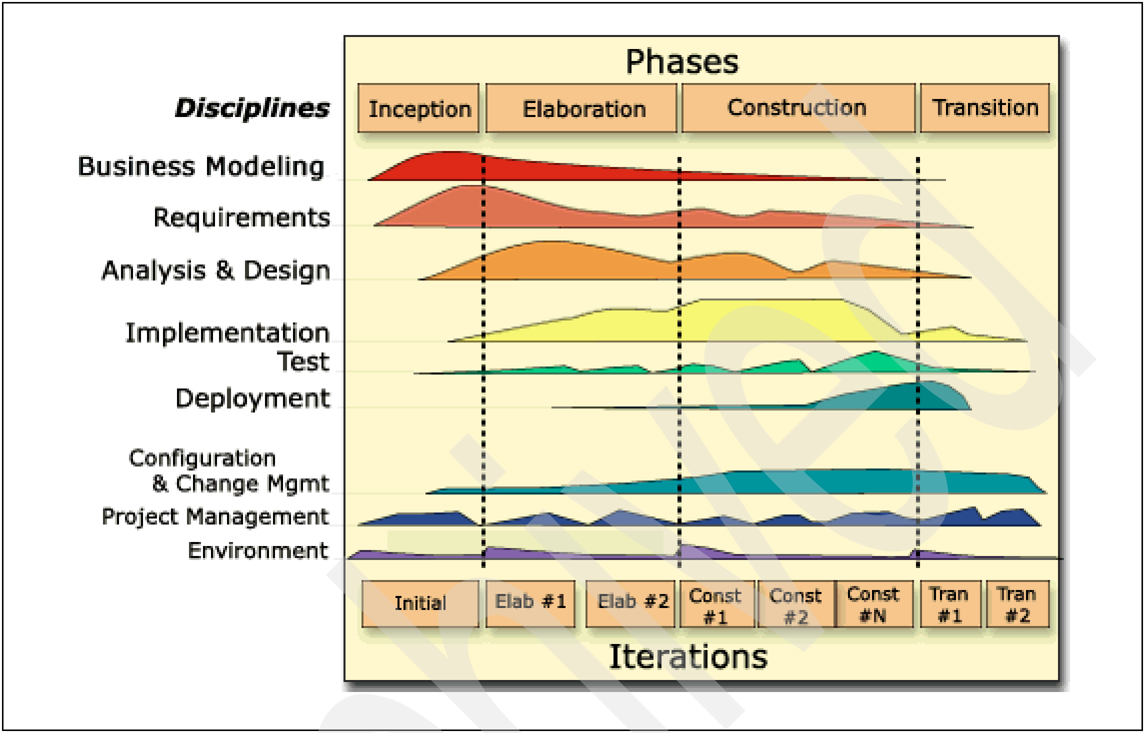


Figure C-2 Architecture of RUP

The "humps" in the chart show the relative emphasis of the disciplines over time. Iterative development is shown; we see all disciplines represented in every iteration. What changes is the emphasis, for example, more requirements in the early stages of the project and more testing in the later stages.

RUP follows a standard meta-model[1] for describing software processes, which includes the following concepts:

- Artifacts: What is produced
- Activities: How to perform the work
- Roles: Who performs the work

[1] See the OMG standard Software Process Engineering Meta-Model, available from http://www.omg.org.

Artifacts

Artifacts can take various shapes or forms, such as:

- A model, such as the use-case model or the design model. These contain model elements (sub-artifacts) such as design classes, use cases, and design subsystems.
- Databases or other types of tabular information repositories such as spreadsheets.
- Source code and executables.
- Various types of documents, for example, a specification document, such as the requirements specification, or a plan document, such as the software development plan.

Roles

A role defines the behavior and responsibilities of an individual, or a set of individuals working together as a team, within the context of a software engineering organization.

Note that roles are not individuals; instead, roles describe responsibilities. An individual will typically take on several roles at one time and frequently will change roles over the duration of the project.

Activities

An activity is work performed by a role. It is usually defined as a series of steps that involve creating or updating one or more artifacts.

Some examples of activities are:

- Find actors and use cases: An activity performed by the system analyst role to identify high-level functional requirements in terms of actors and use cases.
- Describe distribution: An activity performed by the software architect role to describe software distribution across multiple processors.

Workflow and workflow details

A process should describe which activities are performed together and the order in which activities are performed. A workflow is a sequence that shows how work is ordered.

RUP describes a workflow for each discipline. The elements in the workflow are groupings of activities referred to as *workflow details*. The groupings are activities that are performed together to achieve a specific result.

The Patterns for e-business can be used to accelerate the architecture development steps found in the RUP requirements and analysis and design disciplines.

Requirements discipline

The RUP requirements workflow details this discipline in *Analyze the Problem, Understand Stakeholder Needs and Define the System* (see Figure C-3 on page 489). The key goal within the Workflow Details (WKF) is to create a *vision* for the project. The vision contains the Business and IT Drivers (functional and non-functional requirements) and use case models. The requirements discipline is the main discipline in the Inception phase, but with growing experience a higher level of detail is achieved in the Elaboration phase.

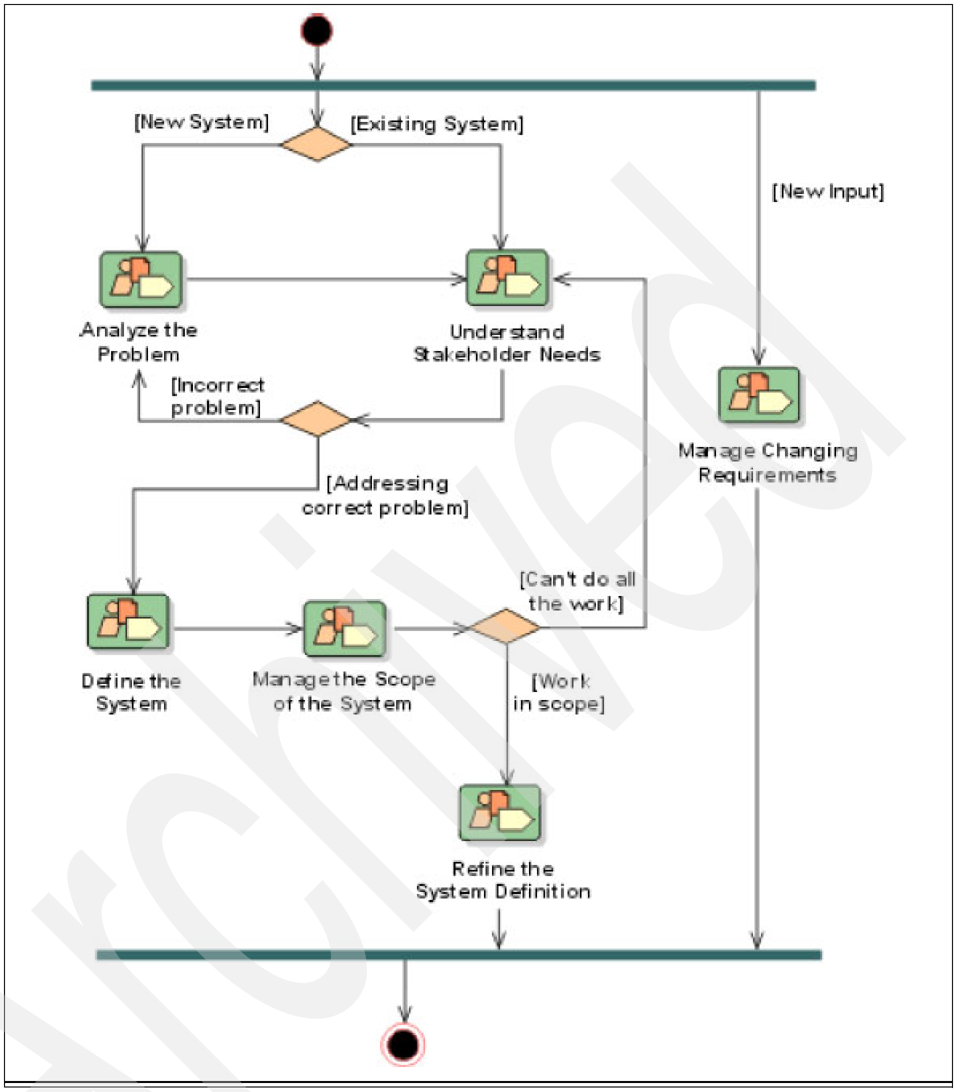
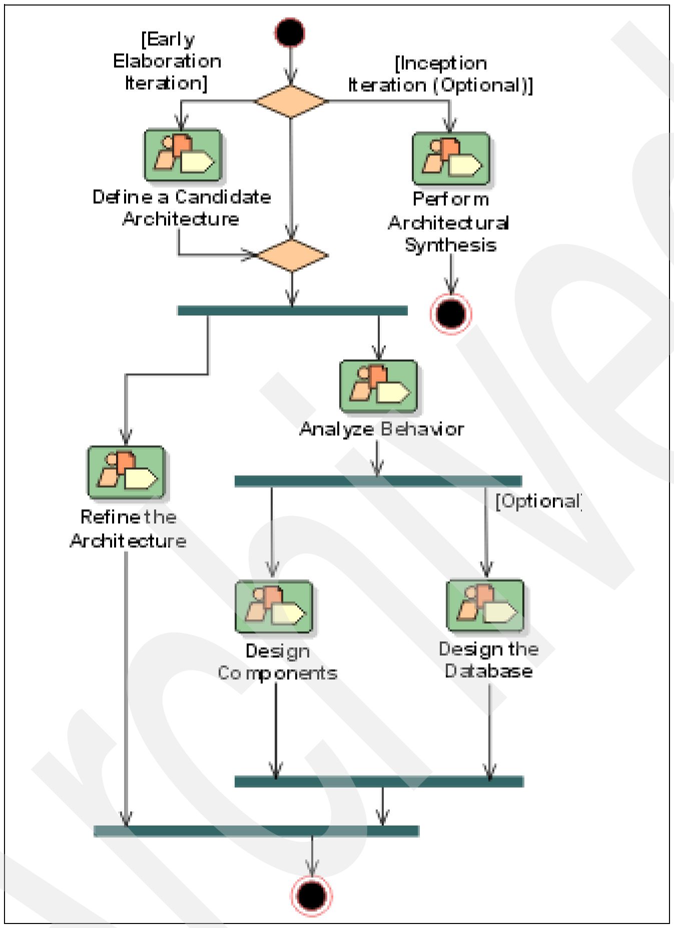

[New System]
[Existing System]
[New Input]
Analyze the Problem
Understand Stakeholder Needs
[Incorrect problem]
Manage Changing Requirements
[Addressing correct problem]
[Can't do all the work]
Define the System
Manage the Scope of the System
[Work in scope]
Refine the System Definition

Figure C-3 Requirements workflow discipline

Analysis and Design discipline

The Analysis and Design discipline contains the creation of a stable architecture, to which the Patterns for e-business can contribute significantly. Although this discipline is optionally performed during the Inception phase, an architectural proof-of-concept (PoC) would be run in the Workflow Detail *Perform Architectural Synthesis* in the Analysis and Design Workflow (see Figure C-4 on page 490). This proof-of-concept could be anything from a documented analysis to a set of throw-away prototypes. The proof-of-concept is recommended because it provides an overall proof that one or more solution patterns are available to match the Business pattern, and it motivates the stakeholders to continue their investment in the project.



[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analyze Behavior
Refine the Architecture
[Optional]
Design Components
Design the Database

Figure C-4 Analysis and design discipline workflow

Candidate architecture

The IT Architect investigates the feasibility of the requirements during early iterations in the Elaboration phase (see Figure C-5 on page 491). A first architectural overview is created, which is ideally based on an existing reference architecture. For this, Patterns for e-business are recommended to identify Application patterns and matching Runtime patterns. Early prototyping is still the recommended approach to reduce risks. These prototypes are near to the solution architecture and are subsets of the future solution rather than throw-away types like in the Inception phase.

The *Refine the Architecture Workflow* Detail is performed with the goal of creating a stable solution architecture. Supplementary requirements are taken into consideration to refine the candidate architecture (for example, non-functional requirements). Architecture significant requirements are detailed, and Patterns for e-business are revisited to confirm that any choices made during proof-of-concept and prototyping work are still applicable. Fit-gap analysis and build/buy decisions determine how much additional work must be invested to complete the architecture.

The diagram shown in Figure C-5 describes the workflow detail "define a candidate architecture."

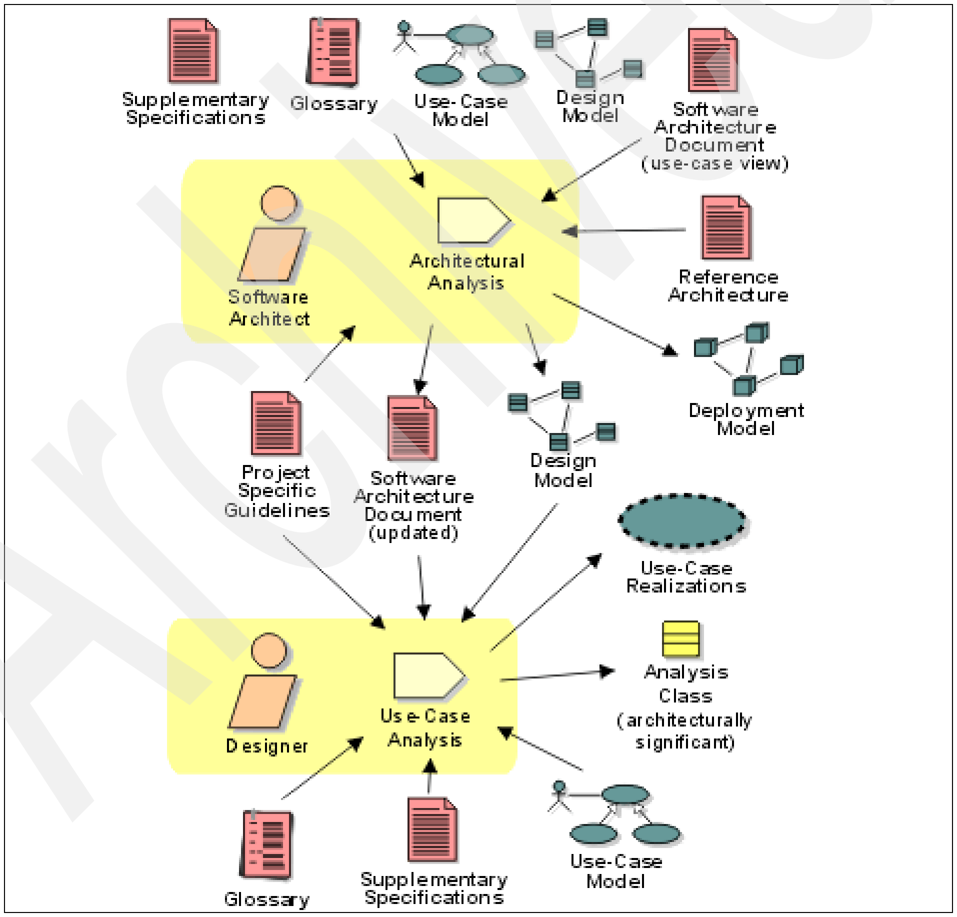

Supplementary Specifications
Glossary
Use-Case Model
Design Model
Software Architecture Document (use-case view)
Software Architect
Architectural Analysis
Reference Architecture
Deployment Model
Project Specific Guidelines
Software Architecture Document (updated)
Design Model
Use-Case Realizations
Designer
Use-Case Analysis
Analysis Class (architecturally significant)
Glossary
Supplementary Specifications
Use-Case Model

Figure C-5 Define a candidate architecture workflow

Table 15-1 provides an overview of the architectural significant activities and artifacts during the Inception and the Elaboration phase, and shows where the Patterns for e-business are relevant.

Table 15-1 Important phases and disciplines for IT Architecture development

RUP discipline	**Activities and Work Flow Details (WFD) in RUP phases**	
	Inception	**Elaboration**
Environment	WFD: Prepare environment for project. Activity: Prepare guidelines for project. * Identification of Patterns for e-Business as potential asset. * Tailor project method (for example, RUP). Artifact: Project-specific guidelines.	Activity: Project environment might be extended. Artifact: Project-specific guidelines (refined).
Business Modelling	Optional WFD: Explore business automation. * Identifying and characterizing of a business process to better understand the business context. Artifact: Business Vision.	Optional Activity: Refinement of Business Vision. Artifact: Business vision (refined).
Requirements	WFD: Define the system. Activity: Develop vision. * Identify Business and IT Drivers, functional and non-functional requirements. * Documentation of business use cases (use case view). * Identify *Business pattern*. Artifact: Vision.	WFD: *Refine* the system definition. Activity: Update/confirm vision. * Detail IT Drivers, functional and non-functional requirements. * Create specific and architectural relevant use case models, functional and logical views. Artifacts: Vision (updated).

RUP discipline	Activities and Work Flow Details (WFD) in RUP phases	
	Inception	**Elaboration**
Analysis and Design	WFD: Perform Architectural synthesis (SF: refer to A and D WFD). Activity: Architectural analysis. * Identify *Application Patterns.* * Select *Runtime Patterns.* * Choose possible products (Product mapping) and derive Deployment View for proof-of-concept. * Prototyping. Artifact: Architectural proof-of-concept * Software/Solution architecture document (inception draft: Architectural Overview).	WFD: Define a candidate architecture. WFD: Refine the solution architecture. Activity: Analyze proof-of-concept. * Fit-gap analysis. * Product evaluation (buy versus develop). * Prioritize (risk and architectural relevance). Artifact: Software/Solution architecture document (Use Case View, Logical View, Implementation View, Deployment View).

Phases

Another key concept in RUP is phases. Phases provide project milestones that ensure that iterations make progress and converge on a solution, rather than iterate indefinitely.

The phases of RUP are:

- Inception: A common understanding of the project lifecycle objectives between the stakeholders is achieved; risks, business and IT requirements are identified.
- Elaboration: The software architecture is established and validated with an executing architectural version of the system.
- Construction: The focus is on completing the development of the system.
- Transition: The software is deployed and made acceptable to its end users.

The objectives of each phase are achieved by executing one or more iterations within the phase. Each phase concludes with a milestone at which the phases are assessed to determine if the project has achieved the specified objectives of the phase. The project cannot move to the next phase until these objectives are achieved. See Figure 15-22 on page 494.

The objectives of each phase are described in more detail in *Using a Single Business pattern with Rational Unified Process*, REDP-3812.

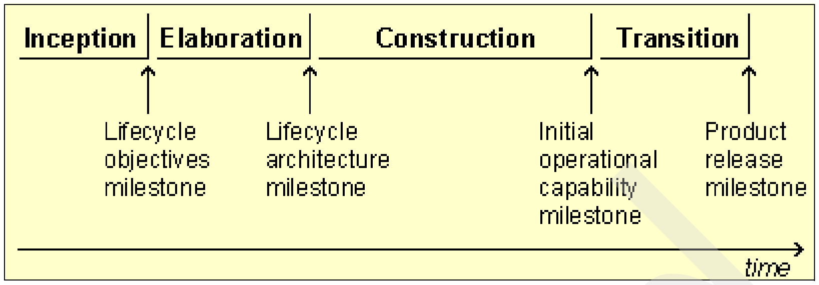

Inception
Elaboration
Construction
Transition
Lifecycle objectives milestone
Lifecycle architecture milestone
Initial operational capability milestone
Product release milestone
time

Figure 15-22 The phases and milestones of a project

For more information about RUP

The following publications can help you learn and master RUP quickly:

- Kruchten, P., *The Rational Unified Process: An Introduction, Second Edition*, Addison-Wesley, 2000, ISBN 0201707101
- Kroll, P. and P. Kruchten, *The Rational Unified Process Made Easy: A Practitioners Guide to the RUP*, Addison-Wesley, 2003, ISBN 0321166094
- Jacobson, I., et al., *The Unified Software Development Process*, Addison-Wesley, 1999, ISBN 0201571692
- Royce, W., *Software Project Management: A Unified Framework*, Addison-Wesley, 1998, ISBN 0201309580
- Many articles in *The Rational Edge* online e-zine, available at:

  `http://www.therationaledge.com`

D

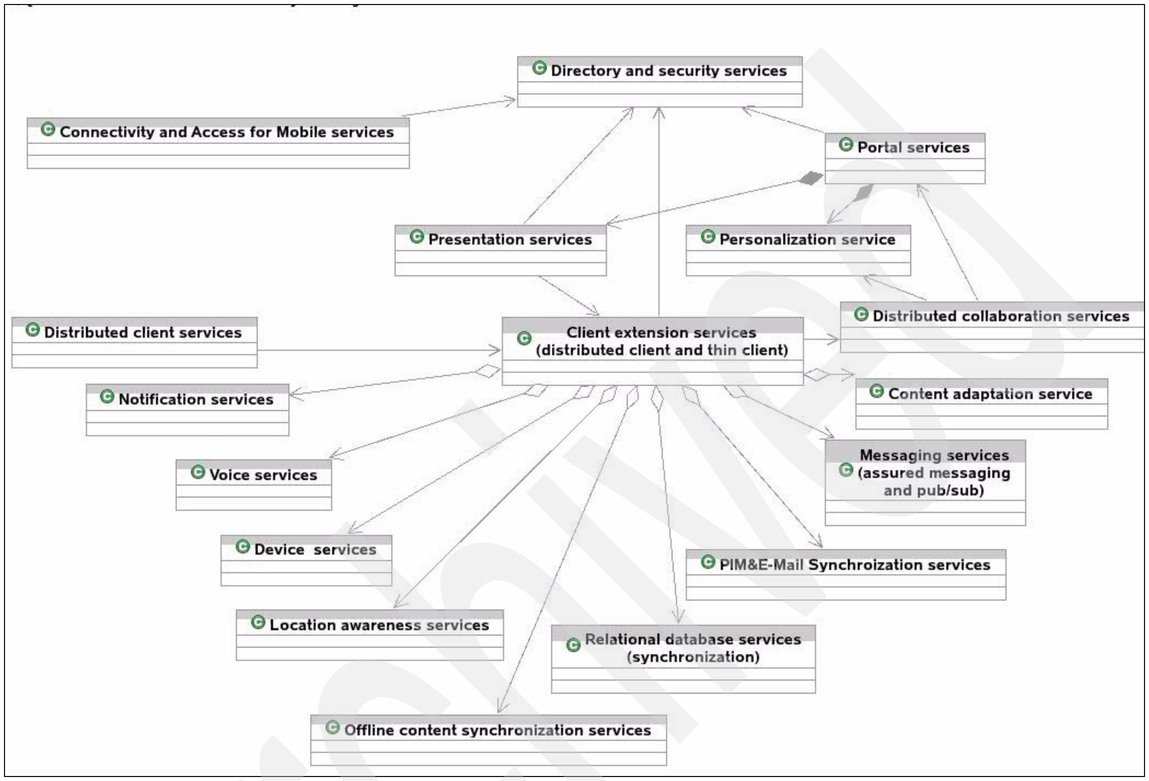
Rational Software Architect component model templates

This appendix highlights the IBM Rational Software Architect component model templates we have included in the ITSO sample code. These assets are meant to be reused by an IT architect to accelerate the component modeling effort.

The Rational Software Architect component model templates can be found in the c:\6775code\model\AccessPattern_ModelPI.zip project interchange file, which can be imported into IBM Rational Software Architect.

Generic component model

Figure D-1 displays the generic component model.

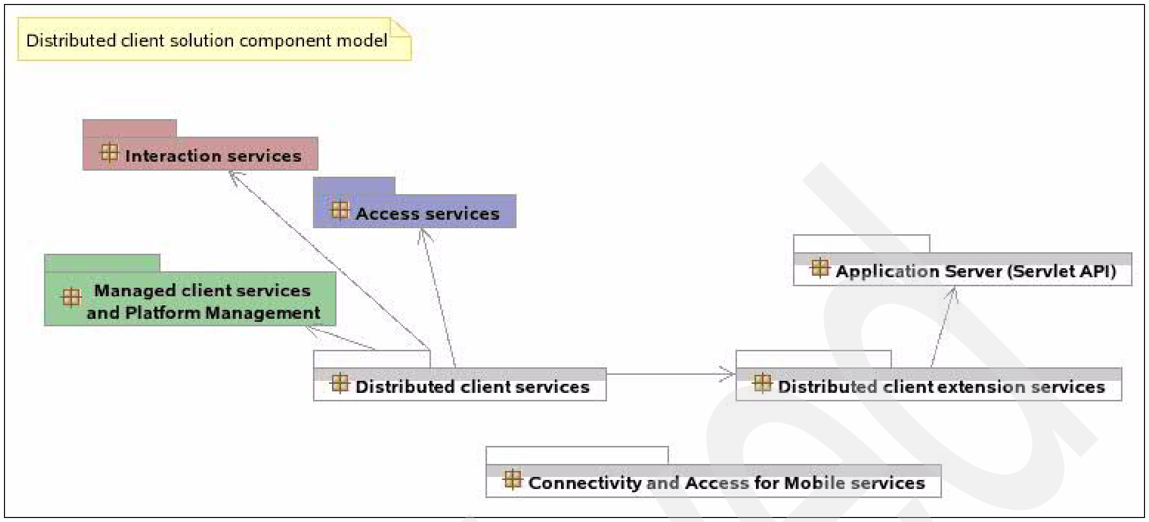


Directory and security services
Connectivity and Access for Mobile services
Portal services
Presentation services
Personalization service
Distributed client services
Client extension services (distributed client and thin client)
Distributed collaboration services
Notification services
Content adaptation service
Voice services
Messaging services (assured messaging and pub/sub)
Device services
PIM&E-Mail Synchroization services
Location awareness services
Relational database services (synchronization)
Offline content synchronization services

Figure D-1 Generic model

Generic implementation model (component view)

Figure D-2 on page 497 displays the generic implementation model (component view).



Distributed client solution component model
Interaction services
Access services
Managed client services and Platform Management
Application Server (Servlet API)
Distributed client services
Distributed client extension services
Connectivity and Access for Mobile services

Figure D-2 Generic implementation model (component view)

Client device model components

Figure D-3 on page 498 displays the client device components.

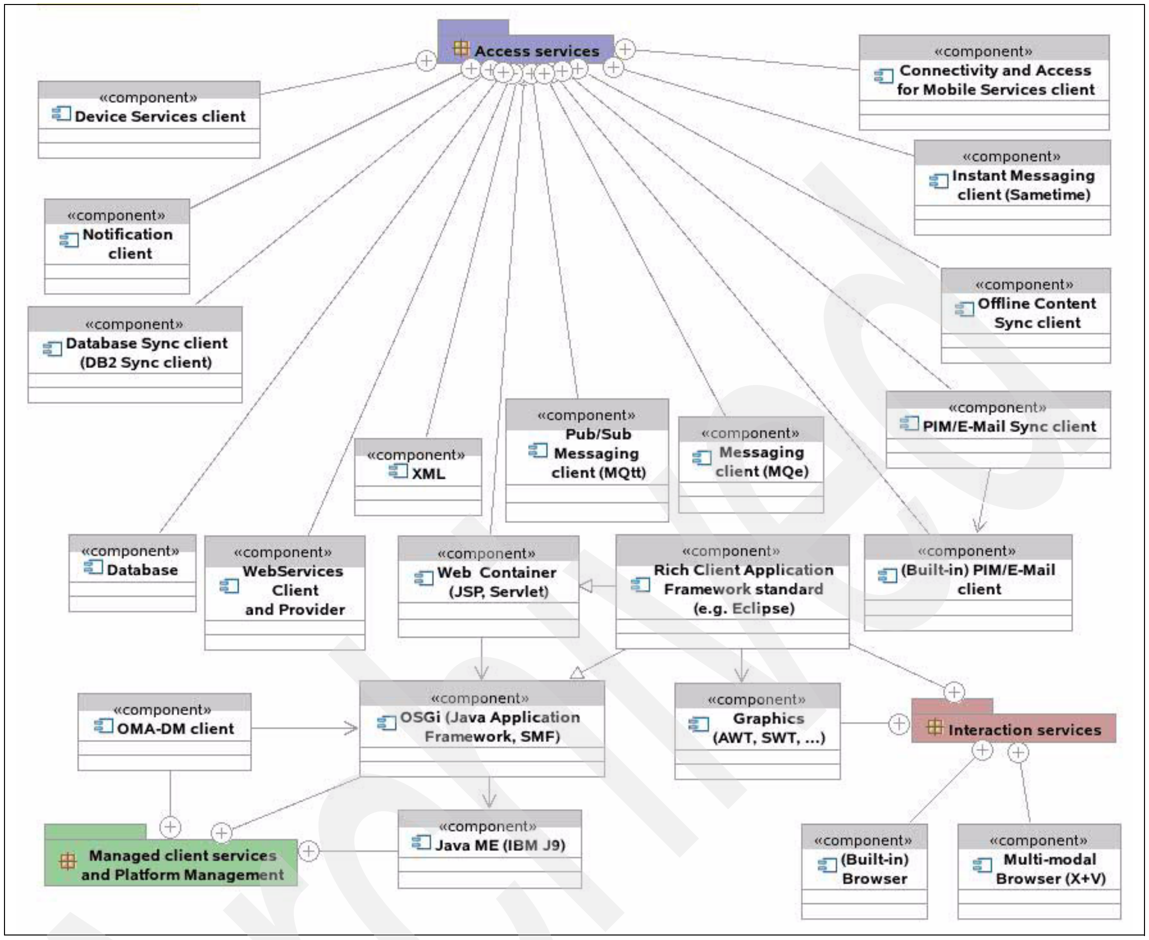

Access services
«component» Connectivity and Access for Mobile Services client
«component» Device Services client
«component» Instant Messaging client (Sametime)
«component» Notification client
«component» Offline Content Sync client
«component» Database Sync client (DB2 Sync client)
«component» PIM/E-Mail Sync client
«component» XML
«component» Pub/Sub Messaging client (MQtt)
«component» Messaging client (MQe)
«component» Database
«component» WebServices Client and Provider
«component» Web Container (JSP, Servlet)
«component» Rich Client Application Framework standard (e.g. Eclipse)
«component» (Built-in) PIM/E-Mail client
«component» OMA-DM client
«component» OSGi (Java Application Framework, SMF)
«component» Graphics (AWT, SWT, ...)
Interaction services
Managed client services and Platform Management
«component» Java ME (IBM J9)
«component» (Built-in) Browser
«component» Multi-modal Browser (X+V)

Figure D-3 Client device components

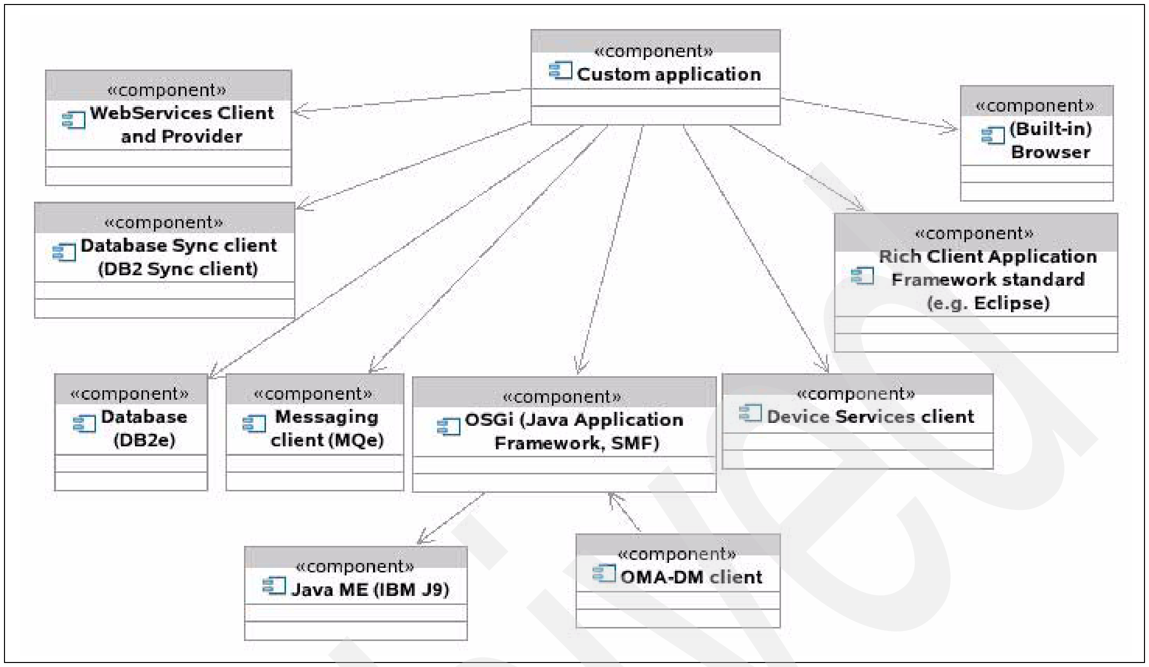


Figure D-4 Client device components for the ITSO working example

Access services components

Figure D-5 on page 500 displays the access services components.

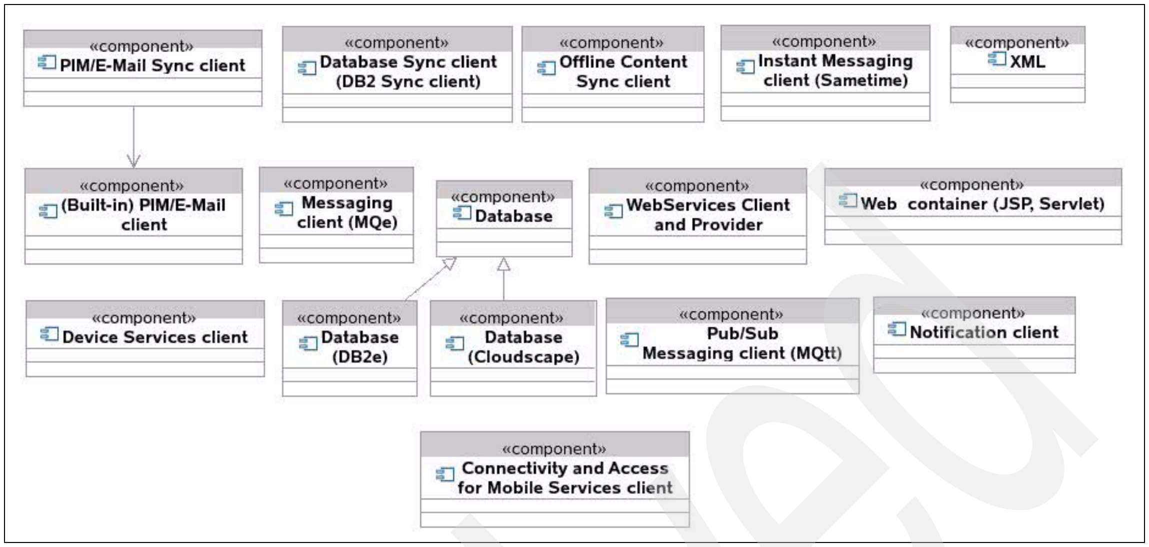

«component» PIM/E-Mail Sync client
«component» Database Sync client (DB2 Sync client)
«component» Offline Content Sync client
«component» Instant Messaging client (Sametime)
«component» XML
«component» (Built-in) PIM/E-Mail client
«component» Messaging client (MQe)
«component» Database
«component» WebServices Client and Provider
«component» Web container (JSP, Servlet)
«component» Device Services client
«component» Database (DB2e)
«component» Database (Cloudscape)
«component» Pub/Sub Messaging client (MQtt)
«component» Notification client
«component» Connectivity and Access for Mobile Services client

Figure D-5 Access services components

Interaction services components

Figure D-6 displays the Interaction services components.

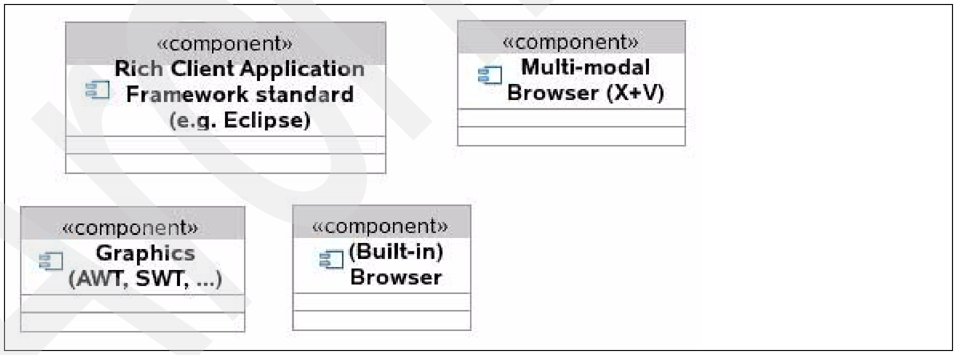

«component» Rich Client Application Framework standard (e.g. Eclipse)
«component» Multi-modal Browser (X+V)
«component» Graphics (AWT, SWT, ...)
«component» (Built-in) Browser

Figure D-6 Interaction services components

Managed client and Platform Management components

Figure D-7 on page 501 displays the managed client and Platform Management components.

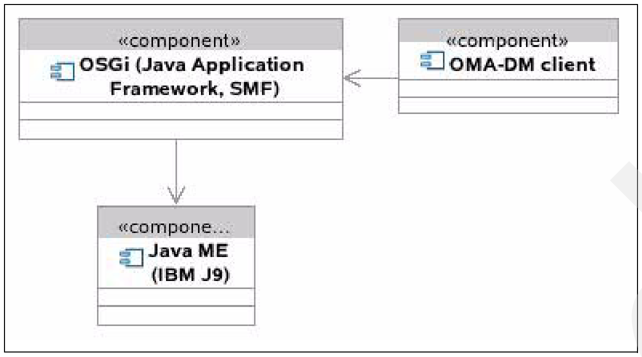


Figure D-7   Managed client and Platform Management components

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/SG246775
```

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG24-6775.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
6775code.zip	Redbook zipped code samples

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	25 MB minimum
Operating System:	Windows
Processor:	1 GHz
Memory:	512 MB

Unzip the 6775code.zip

Where you choose to unzip the 6775code.zip is up to the reader. Throughout the redbook, we will refer to the unzipped contents as the C:\6775code directory.

Description of Web material

Table E-1 provides a brief description of the contents of the 6775code.zip after being unzipped to the C:\6775code directory.

Table E-1 ITSO example application Web download material directory structure

Directory	**Description**
C:\6775code	ITSO example application code root directory.
C:\6775code\deploy\database	ITSO MobileAdjuster database scripts used to create the database, create tables, and load sample data.
C:\6775code\deploy\server	ITSO MobileAdjuster server-side deployable assets.
C:\6775code\deploy\client	ITSO MobileAdjuster client-side deployable assets.
C:\6775code\src\ITSO_MobileAdjusterPI.zip	ITSO MobileAdjuster application source code packaged in a Rational Developer project interchange file.
C:\6775code\model * AccessIntegrationPatterns_model.zip * AccessIntegrationPatterns_ITSO_example.zip	SOA Access Integration patterns reusable assets. ► Rational Software Architect component model templates. ► ITSO Insurance MobileAdjuster working example solution design assets.

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 about ordering these publications, see “How to get IBM Redbooks” on page 508. Note that some of the documents referenced here may be available in softcopy only.

- *Patterns: Pervasive and Rich Device Access Solutions*, SG24-6315
- *Using a Single Business pattern with Rational Unified Process*, REDP-3812
- *Patterns: SOA with an Enterprise Service Bus in WebSphere Application Server V6*, SG24-6494
- *WebSphere Application Server V6 Planning and Design WebSphere Handbook Series*, SG24-6446
- *WebSphere Application Server V6 System Management & Configuration Handbook*, SG24-6451
- *WebSphere Application Server V6 Security Handbook*, SG24-6316
- *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392
- *Rational Application Developer V6 Programming Guide*, SG24-6449
- *WebSphere Version 6 Web Services Handbook Development and Deployment*, SG24-6461
- *WebSphere Everyplace Deployment V6.0 - Client Application Development*, SG24-7183
- *IBM WebSphere Everyplace Deployment V6.0, Installation and Administration*, SG24-7141

Other publications

These publications are also relevant as further information sources:

- *Patterns for e-business: A Strategy for Reuse* by Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos

- *IBM SOA Foundation: An architectural introduction and overview* whitepaper, found at:

  http://www.ibm.com/developerworks/webservices/library/ws-soa-whitepaper
- *System Administrator's Guide, IBM WebSphere Everyplace Deployment V6.0 for Windows and Linux*
- *User's Guide, IBM WebSphere Everyplace Deployment V6.0*
- *Developer's Guide, IBM WebSphere Everyplace Deployment V6.0*
- *Introduction, IBM WebSphere Everyplace MQ*, SC34-6277-02
- *Configuration Guide, IBM WebSphere Everyplace MQ*, SC34-6283-02
- *Application Programming Guide, IBM WebSphere MQ Everyplace*, SC34-6278-01
- *System Programming Guide, IBM WebSphere Everyplace MQ*, SC34-6274-01
- *Application and Development Guide, IBM DB2 Everyplace V8.2*, SC18-7185-04
- Kruchten, P., *The Rational Unified Process: An Introduction, Second Edition*, Addison-Wesley, 2000, ISBN 0201707101
- Kroll, P. and P. Kruchten, *The Rational Unified Process Made Easy: A Practitioners Guide to the RUP*, Addison-Wesley, 2003, ISBN 0321166094
- Jacobson, I., et al., *The Unified Software Development Process*, Addison-Wesley, 1999, ISBN 0201571692
- Royce, W., *Software Project Management: A Unified Framework*, Addison-Wesley, 1998, ISBN 0201309580
- Many articles in *The Rational Edge* online e-zine, available at:

  http://www.therationaledge.com

Online resources

These Web sites and URLs are also relevant as further information sources:

- Patterns for e-business home page

  http://www.ibm.com/developerWorks/patterns
- IBM Education Assistant

  http://www.ibm.com/software/info/education/assistant/WASv60_Task.shtml
- WebSphere Application Server Web site

  http://www.ibm.com/software/webservers/appserv/enterprise

- SUN Java Home page

  `http://java.sun.com`
- IEEE 802 standards

  `http://www.ieee802.org`
- Bluetooth Home page

  `http://www.bluetooth.org`
- World Wide Web Consortium (W3C) HTML Validation Service

  `http://validator.w3.org/`
- World Wide Web Consortium (W3C) information about cHTML (i-Mode)

  `http://www.w3.org/TR/1998/NOTE-compactHTML-19980209`
- World Wide Web Consortium (W3C) information about XForms

  `http://www.w3.org/TR/xforms`
- Open Mobile Alliance Web site

  `http://www.openmobilealliance.org`
- SyncML Official Web Site at:

  `http://www.openmobilealliance.org/tech/affiliates/syncml/syncmlindex.html`
- VoiceXML Forum Official Web site

  `http://www.voicexml.org`
- Microsoft Table PC Web site

  `http://www.microsoft.com/windowsxp/tabletpc`
- Microsoft mobile application development tools Web site

  `http://msdn.microsoft.com/mobility/prodtechinfo/devtools/netcf`
- Microsoft mobile solutions Web site

  `http://www.microsoft.com/mobile`
- OSGI Web site

  `http://www.osgi.org`
- Palm source Web site

  `http://www.palmsource.com`
- Symbian Web site

  `http://www.symbian.com`
- Sun J2ME Web site

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

- Qualcom BREW Web site

  `http://brew.qualcomm.com`
- Sun Java JMS Web site

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

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

IBM

Redbooks

Patterns: SOA Client - Access Integration Solutions

Patterns: SOA Client Access Integration Solutions

Process for reusing the patterns assets within RUP to design a solution

Extend the enterprise by applying Client application patterns

Patterns working example featuring IBM WebSphere Everyplace Deployment V6.0

The Patterns for e-business are a group of proven, reusable assets that can be used to accelerate the design and development of e-business applications. This IBM Redbook focuses on how to reuse the SOA Client - Access Integration patterns to extend enterprise applications to a wide range of client devices such as laptops, PDAs, and mobile phones. Part 1 presents the client access solution domain and benefits of using patterns. We introduce key technologies to serve as a foundation for understanding the Access Integration patterns.

Part 2 guides the reader through the process of reusing the patterns assets within the context of the RUP to accelerate the solution design. We include selection criteria to navigate from the Business and IT drivers to the appropriate Application pattern or patterns, and supporting Runtime pattern, component model, and Product mapping instantiated by IBM software products.

Part 3 demonstrates how to reuse the patterns for a fictitious ITSO Insurance scenario. The scenario highlights the Distributed Rich Client application pattern, which is used to extend the existing SOA enterprise application. The solution provides mobile adjusters with the capability to work on-site with the customer in a disconnectable mode and sync data with the enterprise when connected to the network. The Runtime pattern is instantiated by IBM WebSphere Everyplace Deployment V6.0.

SG24-6775-00 ISBN 0738496421