AT&T Developer Program

Mobile Application Development Quick Start Guide

White Paper

Revision	**1.1**
Revision Date	3/13/2008

This document and the information contained herein (collectively, the "**Information**") is provided to you (both the individual receiving this document and any legal entity on behalf of which such individual is acting) ("**You**" and "**Your**") by AT&T, on behalf of itself and its affiliates ("**AT&T**") for informational purposes only. AT&T is providing the Information to You because AT&T believes the Information may be useful to You. The Information is provided to You solely on the basis that You will be responsible for making Your own assessments of the Information and are advised to verify all representations, statements and information before using or relying upon any of the Information. Although AT&T has exercised reasonable care in providing the Information to You, AT&T does not warrant the accuracy of the Information and is not responsible for any damages arising from Your use of or reliance upon the Information. You further understand and agree that AT&T in no way represents, and You in no way rely on a belief, that AT&T is providing the Information in accordance with any standard or service (routine, customary or otherwise) related to the consulting, services, hardware or software industries.

AT&T DOES NOT WARRANT THAT THE INFORMATION IS ERROR-FREE. AT&T IS PROVIDING THE INFORMATION TO YOU "AS IS" AND "WITH ALL FAULTS." AT&T DOES NOT WARRANT, BY VIRTUE OF THIS DOCUMENT, OR BY ANY COURSE OF PERFORMANCE, COURSE OF DEALING, USAGE OF TRADE OR ANY COLLATERAL DOCUMENT HEREUNDER OR OTHERWISE, AND HEREBY EXPRESSLY DISCLAIMS, ANY REPRESENTATION OR WARRANTY OF ANY KIND WITH RESPECT TO THE INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY REPRESENTATION OR WARRANTY OF DESIGN, PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, OR ANY REPRESENTATION OR WARRANTY THAT THE INFORMATION IS APPLICABLE TO OR INTEROPERABLE WITH ANY SYSTEM, DATA, HARDWARE OR SOFTWARE OF ANY KIND. AT&T DISCLAIMS AND IN NO EVENT SHALL BE LIABLE FOR ANY LOSSES OR DAMAGES OF ANY KIND, WHETHER DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, SPECIAL OR EXEMPLARY, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF GOODWILL, COVER, TORTIOUS CONDUCT OR OTHER PECUNIARY LOSS, ARISING OUT OF OR IN ANY WAY RELATED TO THE PROVISION, NON-PROVISION, USE OR NON-USE OF THE INFORMATION, EVEN IF AT&T HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.

© 2008 AT&T Knowledge Ventures

Revision History

Date	Revision	Description
3/10/2008	1.0	First complete version.
3/13/2008	1.1	Additional vendor-provided information.

Table of Contents

Table of Contents

Tables

Table of Contents

1. Introduction

This paper provides a quick reference for the application-development environments, tools, technologies and libraries associated with developing for mobile platforms. The paper emphasizes RIM BlackBerry, Windows Mobile and Symbian platforms, and also provides information for Palm OS Garnet.

1.1 Audience

The target audience of this white paper is software developers, IT developers, architects, and managers familiar with desktop or server development who are new to mobile development.

1.2 Contact Information

E-mail any comments or questions regarding this white paper via the AT&T Developer Program. Please reference the title of this paper in the message.

1.3 Resources

AT&T Resources

AT&T Developer Program: http://developer.att.com

Mobile Application Development: http://developer.att.com/mobiledevelopment

Device Information: http://developer.att.com/devicedetails

RIM BlackBerry Information: http://developer.att.com/rim

Windows Mobile Information: http://developer.att.com/windowsmobile

Symbian Information: http://developer.att.com/symbian

Java Information: http://developer.att.com/java

Palm Information: http://developer.att.com/palm

Mobile Middleware: http://developer.att.com/middleware

Wireless Reference Architecture Material: http://developer.att.com/WRA

Security Guidelines: http://developer.att.com/security

Certified Application Catalog: http://developer.att.com/certifiedsolutionscatalog

devCentral Resource on Platforms and Operating Systems

1.4 Terms and Acronyms

The following table defines the acronyms used in this document.

Table 1: Terms and Acronyms

Term or Acronym	Definition
2G	Second Generation
3G	Third Generation
Ajax	Asynchronous JavaScript And XML
API	Application Programming Interface
APN	Access Point Name
CIL	Common Intermediate Language
CLDC	Connected Limited Device Configuration
CPU	Central Processing Unit
CRM	customer relationship management
DHCP	Dynamic Host Configuration Protocol
DHTML	Dynamic Hypertext Markup Language
DOM	Document Object Model
EDGE	Enhanced Data Rates for GSM Evolution
eRCP	embedded Rich Client Platform
FLV	Flash Video
GUI	Graphical User Interface
GSM	Global System for Mobile communications
HSDPA	High-Speed Downlink Packet Access
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol over Secure Sockets Layer
I/O	Input/Output
IDE	Integrated Development Environment
IP	Internet Protocol
Java ME	Java Platform Micro Edition
JSON	JavaScript Object Notation

Term or Acronym	Definition
JSR	Java Specification Request
Kbps	Kilobits Per Second
LAN	Local Area Network
Mbps	Megabits Per Second
MIDP	Mobile Information Device Profile
OSGi	*A name, not an acronym*
PDP	Packet Data Protocol
PNG	Portable Network Graphics
PRC	Palm Resource File
RAD	Rapid Application Development
RAM	Random Access Memory
SDK	Software Development Kit
SIM	Subscriber Identity Module
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
VM	Virtual Machine
VPN	Virtual Private Network
WAP	Wireless Application Protocol
WCF	Windows Communications Foundation
WEME	WebSphere Everyplace Micro Edition
WAV	Waveform Audio Format
XML	eXtensible Markup Language

2. Languages and Platforms

For mobile application development there are a variety of languages and platforms to choose from. The following table outlines the main programming languages used for the different mobile platforms and that are discussed in this paper.

Table 2: Primary Languages Used for Mobile Platforms

Platform	C/C++	Java	.NET (C#, VB.NET, etc)
Windows Mobile 5/6	Yes	Yes	Yes
BlackBerry	No	Yes	No
Symbian S60	Yes	Yes	Yes[1]
Palm OS Garnet	Yes	Yes	No
iPhone	Yes	No	No

[1] Via Net60 from RedFiveLabs (Hhttp://www.redfivelabs.comH)

3. Native Code Development Platforms and Tools

3.1 Windows Mobile

Windows Mobile 5/6 development is usually done using the Visual Studio 2005/2008 IDEs (integrated development environments). Since the Visual Studio IDE is mature and used by millions of developers throughout the world, it is a relatively easy platform with which to begin mobile application development.

Below is a summary of the popular developer tools and SDKs available for Windows Mobile development.

Table 3: Visual Studio 2005/2008 Information

Tool	Visual Studio 2005/2008
Description	Popular IDE created by Microsoft that can be used for desktop, Web, and mobile application development.
Licensing	Can be purchased separately, as part of an MSDN subscription, or as part of a volume license purchase.
Installation	Standard Windows GUI installation and configuration.
Typical use	Development occurs on a Windows based PC on which Visual Studio is installed. Initial debugging occurs on the same PC using the provided Windows Mobile emulator. Binaries and resources are deployed to the device using Visual Studio or ActiveSync. Logging and remote debugging are then performed on the device to validate application operation.
URLs	http://msdn2.microsoft.com/en-us/vs2005/default.aspx http://msdn2.microsoft.com/en-us/vs2008/default.aspx

Table 4: Windows Mobile 6 SDK

Tool	Windows Mobile 6 SDK
Description	Set of libraries and tools used to build applications for Windows

Tool	Windows Mobile 6 SDK
	Mobile 6 devices.
Licensing	Free download, Microsoft terms and conditions apply.
Installation	Standard Windows GUI installation and configuration.
Typical use	This SDK is typically used in conjunction with Visual Studio to build Windows Mobile 6 applications. It can also be used with other compilers and toolkits.
URL	http://www.microsoft.com/downloads/details.aspx?familyid=06111a3a-a651-4745-88ef-3d48091a390b&displaylang=en

Table 5: Windows Mobile 5 SDK

Tool	Windows Mobile 5 SDK
Description	Set of libraries and tools used to build applications for Windows Mobile 5 devices.
Licensing	Free download, Microsoft terms and conditions apply.
Installation	Standard Windows GUI installation and configuration.
Typical use	This SDK is typically used in conjunction with Visual Studio to build Windows Mobile 5 applications. It can also be used with other compilers and toolkits.
URL	http://www.microsoft.com/downloads/details.aspx?familyid=83A52AF2-F524-4EC5-9155-717CBE5D25ED&displaylang=en

Table 6: Microsoft SQL Server CE Tools for Visual Studio 2005/2008

Tool	Microsoft SQL Server CE tools for VS 2005/2008
Description	Toolset that can be used to assist in mobile applications that use SQL Server CE.
Licensing	Free download, Microsoft terms and conditions apply.

Tool	Microsoft SQL Server CE tools for VS 2005/2008
Installation	Standard Windows GUI installation and configuration.
Typical use	Installs a design-time environment that enables developers to create database applications that use SQL Server Compact Edition. Typically used for mobile applications that need to cache or edit portions of an enterprise database.
URL	http://www.microsoft.com/downloads/details.aspx?FamilyID=877c0adc-0347-4a47-b842-58fb71d159ac&displaylang=en

For more information, visit http://developer.att.com/windowsmobile

3.2 .NET Compact Framework

In addition to C/C++ development for Windows Mobile devices, another popular option is to develop applications for the .NET Compact Framework. The .NET CF is a version of the .NET Framework ported to operate under the constraints of Windows CE, Pocket PC, Smartphone and Windows Mobile operating systems. The .NET CF consists of a runtime system similar to the .NET Compact Framework, as well as the supporting class libraries and tools required to utilize the runtime. In this regard it is very similar to Java. Unlike Java, however, it is designed to be language agnostic and simply specifies the CIL (common intermediate language) instructions. Therefore all of the .NET supported languages, C#, VB.NET, J#, and many others, compile to the same CIL and can be executed by the .NET CF runtime.

The following table shows the versions of .NET CF, the versions of Windows Mobile supported, and the features contained in each.

Table 7: .NET Compact Framework Details

.NET CF Version	WM Version	Visual Studio Version Supported	Main Features
1.0	5.0 (in ROM)	2005	Basic WinForms classes, Web services, Bluetooth, IrDA, and much of the core .NET platform.

.NET CF Version	WM Version	Visual Studio Version Supported	Main Features
2.0	5.0 (user installable) 6.0 (in ROM)	2005, 2008	UI enhancements, XPath support, SOAP 1.2, MSMQ, serial port access, cryptography and performance enhancements, background threads
3.5	6.0 (user installable)	2008	WCF (Windows Communications Foundation), LINQ (language-integrated , set and transform operations), many WinForms enhancements, SoundPlayer, compression, .NET CF profiler, and platform ID support

For more information, visit http://developer.att.com/windowsmobile

3.3 Symbian

There are several choices currently available for Symbian application development; however, the most commonly used tools are provided by Nokia and are part of the Carbide set of products and tools. At this time there are two main IDEs used for Symbian development. The first is Microsoft's Visual Studio used in conjunction with Carbide.vs and the second is the Carbide.c++ Eclipse-based development environment. These powerful IDEs make it easy to develop full-featured applications for S60 devices.

Table 8: Carbide.c++

Tool	**Carbide.c++**
Description	The Carbide.c++ product is a stand-alone platform based on Eclipse.
Licensing	Nokia terms and conditions apply. However, it is available as a free developer version or for purchase in Pro and OEM versions.

Tool	Carbide.c++
Installation	Standard Windows GUI installation and configuration. Although it is an Eclipse-based application, it only runs on Windows platforms.
Typical use	Carbide.c++ is used in conjunction with at least one of the Symbian OS SDKs in order to create applications for Symbian OS based devices.
URL	http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide_cpp/

Table 9: Carbide.vs

Tool	Carbide.vs
Description	Toolset that integrates with Visual Studio 2003 and 2005.
Licensing	Nokia terms and conditions apply. However, it is available as a free developer version or for purchase in Pro and OEM versions.
Installation	Standard Windows GUI installation and configuration.
Typical use	Carbide.vs plugs into Visual Studio 2003 and 2005 in order to allow application development for the Symbian OS in the familiar Visual Studio environment.
URL	Version 2.0.2 - http://www.forum.nokia.com/info/sw.nokia.com/id/9124f1f0-1fc0-405e-9c60-facf7b337702/Carbide_vs_2_0_1.html Version 3.0.1 - http://www.forum.nokia.com/info/sw.nokia.com/id/a133232e-8a5a-4bd3-973e-a48edeb098b0/Carbide_vs_3_0.html

Table 10: Carbide.ui

Tool	Carbide.ui

Tool	Carbide.ui
Description	Symbian theme manager toolkit.
Licensing	Free download, but Nokia terms and conditions apply.
Installation	Standard Windows GUI installation and configuration.
Typical use	The Carbide.ui Theme Edition allows developers to create themes for S60 and Series 40 devices. It gives developers access to all of the customizable elements in S60 and S40 user interfaces.
URL	http://www.forum.nokia.com/info/sw.nokia.com/id/bb173537-4e67-496f-9967-50917d5cfc47/S60_Theme_Studio_for_Symbian_OS.html

Table 11: Open C SDK Plug-In for S60

Tool	Open C SDK Plug-In for S60
Description	Provides access to POSIX and other open standard compatibility APIs.
Licensing	Free download, but Nokia terms and conditions apply.
Installation	Standard Windows GUI installation and configuration.
Typical use	The Open C SDK allows developers to create applications that conform to POSIX specifications and other open-standards APIs, enabling applications to leverage and share existing code with other projects.
URL	http://www.forum.nokia.com/info/sw.nokia.com/id/91d89929-fb8c-4d66-bea0-227e42df9053/Open_C_SDK_Plug-In.html

Table 12: WSDL to C++ Wizard

Tool	WSDL to C++ Wizard
Description	Allows developers to quickly create C++ code that can call web

Tool	WSDL to C++ Wizard
	services.
Licensing	Free download, but Nokia terms and conditions apply.
Installation	Standard Windows GUI installation and configuration.
Typical use	The wizard allows users a Visual Studio to quickly create C++ proxy stubs by simply specifying the WSDL file for the Web service. In addition it provides a command-line tool.
URL	http://www.forum.nokia.com/info/sw.nokia.com/id/5ddeb939-c4e4-4e64-8f25-282e1e86afed/Nokia_WSDL_to_Cpp_Wizard_for_S60.html

Table 13: Coverity Prevent

Tool	Coverity Prevent v3.6
Description	Static analysis tool to detect defects.
Licensing	Commercial terms from Coverity.
Installation	Standard Windows GUI installation and configuration.
Typical use	Allows developers to easily find defects in their code.
URL	http://www.coverity.com/

Table 14: Klockwork

Tool	Klocwork
Description	Static analysis tool to detect defects.
Licensing	Commercial terms from Klocwork.
Installation	Standard Windows GUI installation and configuration.

Tool	Klocwork
Typical use	Allows developers to easily find defects in their code.
URL	http://www.klocwork.com

Table 15: EUnitPro

Tool	EUnitPro
Description	EUnit Pro enables creation and automation of unit and module tests for Symbian C++ Software.
Licensing	Commercial terms from SysOpen Digia.
Installation	Standard Windows GUI installation and configuration.
URL	http://www.sysopendigia.com/C2256FEF0043E9C1/0/405001346

Table 16: Bullseye Coverage

Tool	Bullseye Coverage
Description	Code coverage tool.
Licensing	Commercial terms from Bullseye
Installation	Standard Windows GUI installation and configuration.
URL	http://www.bullseye.com/

For more information, visit http://developer.att.com/symbian

3.4 Symbian Editions

Symbian is both a platform and an operating system. Although there are many Symbian platforms, including S60, S80 and UIQ, this paper emphasizes the S60 platform. The platform itself is both a specification of the OS version and the form factor for supported devices.

Table 17: S60 Versions

Edition	OS Version	Major Features
1st	6.1	PIM, RealPlayer, XHTML Mobile Profile
2nd	FP1 - 7.0 FP2 – 8.0a FP3 – 8.1a	Multi-homing, MIDP 2.0, ECom plug-in framework, EDGE, lightweight multithreaded multimedia framework, WCDMA, and WAP 2.0 *Feature Pack 1* – Megapixel camera w/ 4x zoom and video playback *Feature Pack 2* – 1.3 Megapixel camera w/ 6x zoom, WCDMA, EDGE, IPv6 *Feature Pack 3* – Scalable UI support (multiple screen sizes)
3rd	9.1 FP1 – 9.2	New EPOC kernel, New ABI ARM compiler, Enhanced security and many OMA DRM 2.0 features. *Feature Pack 1* – Open source browser, optional Flash 2.0, and enhanced UI.

3.5 Palm OS Garnet

Palm OS application development options are much more diverse than Windows Mobile or Symbian. In addition to the IDE made available by Access, there is also the CodeWarrior IDE and a set of command-line tools.

Table 18: PRC-Tools

Tool	PRC-Tools
Description	Open source collection of tools for C/C++ Palm OS development.
Licensing	GNU open source license.
Installation	Requires Linux x86 OS or Cygwin on Windows. Also requires installation of a Palm OS SDK.
Typical use	These tools allow developers to create Palm OS based applications on a variety of operating systems. It includes command-line tools to compile, link and build PRCs as well as an FLTK based emulator and a GTK based Palm resource compiler. The resource compiler is used to create the UI portions of the application much like can be done in a standard IDE.
URL	http://prc-tools.sourceforge.net/

Table 19: CodeWarrier for Palm OS

Tool	CodeWarrior for Palm OS
Description	The CodeWarrior IDE and toolset for the Palm OS.
Licensing	Licensed by Freescale.
Installation	Standard Windows GUI installer.
Typical use	This IDE allows developers to create Palm applications in RAD environment without needing in-depth command-line knowledge as with the PRC-Tools. It is used in conjunction with a Palm OS SDK to design, build, and debug Palm OS applications.
URL	http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CWX-PLM-PROED-CX&nodeId=01272694018472

Table 20: Access Garnet OS Tools

Tool	Access Garnet OS Tools
Description	Access Garnet OS Tools
Licensing	Free download. Access terms and conditions apply
Installation	Standard Windows GUI installer. Although this contains an Eclipse-based IDE, installation is only available for Windows platforms.
Typical use	This is the set of tools and SDKs distributed by Access for use in developing Palm OS applications. The major component in this toolset is the Garnet OS Developer Suite. This is an Eclipse-based development environment which allows developers to quickly create Palm applications in a RAD environment. In addition, the toolset includes the core Garnet OS SDK, a Palm emulator for debugging on the PC and a web-browser SDK.
URL	http://www.access-company.com/developers/downloads/palmostools.html#link1

For more information, visit http://developer.att.com/palm

4. Java Mobile Application Development

4.1 History

PersonalJava was the first commercially available Java platform for mobile and embedded devices and was nearly equivalent to Java 1.1.8 in features and libraries. J2ME (Java 2 Micro Edition) superseded PersonalJava for use in mobile and embedded devices. J2ME was based upon the Java 2 platform and developed under the Java Community Process as JSR (Java Specification Request) 68. Most recently, J2ME has been renamed Java ME for Java Platform, Micro Edition.

4.2 Java Specification Request

A Java Specification Request or JSR is the name given to the open and standardized process of creating APIs to fill a particular need or provide specific functionality. These JSRs are created by working groups. Some of the core members who create and review JSRs for mobile development are representatives from companies such as Nokia, Sony, Apache Software Foundation, and of course Sun Microsystems. JSRs can be in a state of flux until they are finalized, consisting of an initial review, early draft review, public review, and proposed final draft stage. For example, MIDP 3 or JSR 271 is in the public review stage, meaning it should be in the final release stage shortly. The following is a list of some of the JME related JSRs:

Table 21: Some Important JSRs

JSR	Purpose
JSR 75	For accessing PIM data and for accessing file systems
JSR 82	Java APIs to allow Java-enabled devices to integrate into a Bluetooth environment
JSR 118	Defines MIDP 2.0 and the core APIs that are required by devices conforming to MIDP 2.0
JSR 120	Defines support for SMS, cell broadcast and other cellular related messaging services

JSR	Purpose
JSR 135	Defines the mobile media APIs for working with audio and video
JSR 139	Defines CLDC (Connected Limited Device Configuration)1.1 which specifies the core framework required for small mobile limited connection devices
JSR 172	Defines the Web services specification that enable Java ME clients to consume Web services
JSR 179	Defines the location API that for applications to retrieve location information via GPS or other methods
JSR 232	Defines the OSGi[2] Service Platform which allows for mobile device management and an embedded Rich Client Platform (eRCP)
JSR 248	Creates a mobile-service architecture and platform definition for the high volume wireless handsets continuing the work started in JSR-185
JSR 249	Defines the next generation of the Mobile Service Architecture and platform created in JSR 248

4.3 Profiles and Configurations

Java ME is logically componentized into profiles and configurations that can be difficult for new Java mobile developers to understand. A profile is a set of APIs, designed for a specific configuration, that address the needs of a narrow device category. The most popular Java ME profile is MIDP or Mobile Information Device Profile. This profile is designed for use in devices such as smartphones, PDAs, and other phone-like devices. A configuration combines a Java virtual machine and APIs that are shared across a class of devices. CLDC or Connected Limited Device Configuration is the configuration required by MIDP 2.0. For more information on MIDP refer to JSR 118 and for more information on CLDC refer to JSR 139. In addition to CLDC there is CDC (Connected Device Configuration) which is more feature-rich than CDC and closer to Java SE. The

[2] OSGi is a name and not an acronym

latest version of CDC, version 1.1.2, contains full file I/O support, weak references, reflection, and JNI (Java Native Interface) functionality.

4.4 OSGi Service Platform

Another Java-related technology that is seeing adoption is the OSGi Service Platform. This is a specification for a platform that provides software component management, remote component management such as via SNMP, application cooperation, and simplified deployment. OSGi implementations require a minimum CDC 1.1 profile and supporting JVM. IBM's Lotus Expeditor is an example of an OSGi Service Platform implementation. Using Expeditor, developers can quickly and easily create, deploy and manage mobile java applications. Expeditor can assemble applications from custom portlets, Web applications, Web services, and custom built eRCP applications.

For more information, see:

http://www.osgi.org/Specifications/HomePage

http://publib.boulder.ibm.com/infocenter/ieduasst/lotv1r0/topic/com.ibm.iea.expeditor_v6/expeditor/6.1/ClientPlatform/Client_Device_platform_overview.pdf

4.5 Java VMs

One of the things to consider when you develop a Java application is what type of Java support is available for the device. Fortunately, on many devices and smartphones, Java support is built right in; however, with Windows Mobile you will have to either instruct your users on how to install a suitable virtual machine, or include it as part of your program's installation. The table below lists the supported VMs for the various platforms.

Table 22: Common Java Virtual Machines

Java VM	Platforms	License
IBM WEME J9	Windows Mobile and Palm OS Garnet	IBM Proprietary, per copy fee & license
Mysaifu JVM	Windows Mobile	GPLv2

Java VM	Platforms	License
Symbian JVM	Symbian S60	Symbian, preinstalled
BlackBerry JVM	Blackberry devices only	RIM, preinstalled

Symbian and BlackBerry platforms have a JVM pre-installed. For BlackBerry, Java is the primary programming approach . For Windows Mobile, there are several other JVMs not listed. IBMs WEME (WebSphere Everyplace Micro Edition) is available for both Windows Mobile and Palm OS Garnet devices, but at a per-device cost. For an enterprise, this means you will have to purchase a client license for every device that is required to run your application and if you are a third party software developer, you will need to bundle the JVM cost into your application or instruct users to separately purchase and install it. The Mysaifu JVM is available for Windows Mobile and is free. If it supports your application properly, you can bundle it or instruct users to separately download and install it.

4.6 Popular Java IDEs

Java ME developers can choose from a variety of modern feature-rich IDEs. Although, this list is not inclusive it includes many of the most popular IDEs.

Table 23: Java IDEs

IDE	Visual MIDP Designer?	Developer OSs	License
BlackBerry JDE	No	Windows	RIM License[3]
Eclipse JDT	No	Windows, Mac OS X, Linux, Solaris	Eclipse Public License
NetBeans 6.0.1	Yes	Windows, OS X, Linux, Solaris	GPL2

[3] No fee charged for BlackBerry JDE and many other IDEs

IDE	Visual MIDP Designer?	Developer OSs	License
IntelliJ IDEA 7.0	No	Windows, OS X, Linux	JetBrains License
WebSphere Everyplace Micro Environment	No	Windows, OS X, Linux, Solaris	IBM License

4.7 Installation

Due to the large number of mature and feature-rich IDEs, the process for setting up for and beginning mobile application development with Java is an easy process. Below we enumerate the steps required to install and configure the developer tools.

4.7.1 Eclipse Setup

1. Download and install the Java 2 SE SDK. http://java.sun.com/javase/downloads/index.jsp

2. Download and install Sun's Java Wireless Toolkit 2.5.2. http://java.sun.com/products/sjwtoolkit/download.html?feed=JSC

3. Download and install the Eclipse IDE. http://www.eclipse.org/downloads/

4. From within Eclipse, install the EclipseME toolkit.

    a. Add the EclipseMe remote update site to the list that appears: http://eclipseme.org/updates

    b. When you browse the EclipseME update site from this point, you should see the EclipseME feature. Install it from here.

    c. You can now open the "Preferences" dialog within Eclipse and you should see a "J2ME" item which will allow you to customize your mobile development environment.

5. Configure the EclipseME feature within Eclipse. You will need to import at least one device definition from the wireless toolkit that you downloaded in step 2. This can be done from the Eclipse Preferences dialog under the J2ME item.

4.7.2 NetBeans Setup

1. Download and install the NetBeans IDE 6.0.1 installer which includes all components. You can also use the Mobility version but it does not include the Web Services Connection wizard.
   http://download.netbeans.org/netbeans/6.0/final/

2. The NetBeans installation includes the Wireless Toolkit and everything you need to immediately start with Java mobile development.

4.7.3 IntelliJ IDEA 7.0

1. Download and install Sun's Java Wireless Toolkit 2.5.2.
   http://java.sun.com/products/sjwtoolkit/download.html?feed=JSC

2. Download and install IntelliJIDEA 7.0.
   http://www.jetbrains.com/idea/download/index.html

4.7.4 WebSphere Everyplace Micro Environment

This environment is actually a plug-in to the Eclipse IDE as opposed to being a stand-alone IDE. However, it allows access to IBM's J9 JVM which has more features than the standard Java ME specified JVM, but is still fully compatible with MIDP 2.0 and CLDC 1.1. The JVM is also available for all BREW, Palm OS XScale, and Windows Mobile devices. You will also need to contact an IBM representative in order to receive a trial version of WEME for the platform you are interested in developing for. The installation is straightforward.

1. Install the Eclipse IDE as previously outlined.

2. Install the WebSphere Everyplace Micro Environment.

4.7.5 BlackBerry JDE

The BlackBerry JDE or Java Development Environment is a tool specifically built by Research in Motion (RIM) for creating Java applications for the BlackBerry platform. It contains a number of sample projects to get started. In addition it contains valuable reference documentation for BlackBerry's Java ME platform.

Although RIM's Java ME implementation is MIDP 2.0 and CLDC compliant, it also contains additional APIs for working specifically on BlackBerry devices. These include additional BlackBerry-specific UI APIs, MDS APIs, and all of the other "net.rim.*" APIs for accessing all of the functionality available on BlackBerry devices. For example, there are APIs to access BlackBerry specific hardware or features that JME does not support.

The JDE can be downloaded directly from RIM's website and installation is simple and straight-forward.

1. Download and install the Java 2 SE SDK.
   http://java.sun.com/javase/downloads/index.jsp

2. Download and install the BlackBerry JDE. (Note: It is important that you determine the lowest version of BlackBerry handheld code for which you are developing, and download the corresponding version of the JDE. The JDE is not backward compatible; however applications are forward compatible.)
   http://na.blackberry.com/eng/developers/downloads/jde.jsp

4.8 Use Case Considerations

One of the key advantages to Java application development for mobile devices is that you can target a wide swath of devices with a single source-code base. As long as the devices you are targeting support the MIDP and CLDC versions you are using, then your compiled code should work equally on all of them. One drawback is that the devices you intend to target must support the MIDP and CLDC versions that you require, and that Java provides the capabilities you need. Otherwise you will have to determine if there are work arounds or if a completely different development path like C++ might be better. For instance, if you require access to new hardware features that are not exposed in the MIDP APIs, you will most likely have to perform application development with C++.

Another factor you should consider is performance. Although the mobile Java virtual machines are very high performance, they are still not going to be as fast as a well written C++ application. Therefore, if your application is heavily computational, such as encryption, image manipulation, voice recognition, etc., you may have to develop in C++ to achieve the performance you desire.

4.9 Third-Party Libraries

One important part of development is being aware of some of the third party libraries. The table below lists several of these, but there are many others and in combination with the supported JSRs, you can create full-featured mobile applications.

Table 24: Examples of Java Third-Party Libraries

Library	License	Purpose
Tasman Barcode Reader	Tasman License	Allows your Java MIDlet to read and process barcodes.
McObject Perst	Open source and commercial	Allows your Java MIDlet to utilize the power of an ACID compliant object database.
Wingfoot SOAP Client	Open source	Allows memory constrained devices to use Web services.

For more information, visit http://developer.att.com/java

5. Libraries and Technologies

Due to the special constraints and capabilities of mobile devices, many of the available libraries and APIs may not be familiar to desktop developers. In particular, areas involving communications, whether they are data or voice, personal information management, SIM access, and application deployment will be different in the mobile world or non-existent in the desktop context. The following sections highlight many of the key libraries and APIs that are unique to mobile application development. These can be grouped into the following sections.

1. Data Communications and Connections
2. Voice and Phone Operations
3. SMS sending and receiving
4. Security
5. Installation and Deployment
6. Data Communications and Connections

This section covers the APIs and libraries available on each of the platforms for using the various types of available connections.

Table 25: Windows Mobile 5 and 6 C/C++ Libraries

Library	Purpose	URL
C/C++ Connection Manager	For devices to automatically determine and create best connection based on hardware and user settings.	http://msdn2.microsoft.com/en-us/library/bb416564.aspx
Cell Core	SMS, WAP, SIM and radio interface management.	http://msdn2.microsoft.com/en-us/library/aa921520.aspx
VoIP Services	For applications to place and receive SIP calls.	http://msdn2.microsoft.com/en-us/library/aa926533.aspx
Wireless	For Wi-Fi, IR, and Bluetooth	http://msdn2.microsoft.com/en-

Networking APIs	communications.	us/library/aa915901.aspx
Remote Access Services	Creation of VPN and PPP connections.	http://msdn2.microsoft.com/en-us/library/aa918513.aspx
Remote API 2	For desktop applications to perform operations on a Windows Mobile device.	http://msdn2.microsoft.com/en-us/library/aa920150.aspx
Networking Core	APIs for direct use of Windows sockets, Berkeley sockets.	http://msdn2.microsoft.com/en-us/library/aa917156.aspx
Security	For storing certificates and performing cryptographic functions.	http://msdn2.microsoft.com/en-us/library/bb416393.aspx
Installation	For creating a delivery CAB file that contains all application resources and binaries. VS 2005/2008 have built-in wizards to automatically create an installer for you.	http://msdn2.microsoft.com/en-us/library/bb158621.aspx

Table 26: Windows Mobile 5 and 6 .NET CF Libraries

Library	Purpose	URL
Microsoft.WindowsMobile.Telephony	For applications to place voice calls.	http://msdn2.microsoft.com/en-us/library/microsoft.windowsmobile.telephony.aspx
Microsoft.WindowsMobile.PocketOutlook	For sending of SMS and managing PIM data and other functions.	http://msdn2.microsoft.com/en-us/library/microsoft.windowsmobile.pocketoutlook.messageinterception.aspx

Table 27: Symbian S60 Libraries

Library	Purpose	URL
RComm	For transfer of data over serial interfaces such as USB.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/SerialCommsServerClient/RCommClass.html
Connection Manager and Generic Connection Manager	For opening and closing connections generically.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/ConnectionManager/RConnectionClass.html#%3a%3aRConnection http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/GenConn/index.html#NifManAgents%2etoc
Bluetooth APIs	APIs to handle Bluetooth functions and security as well as the standard Bluetooth UI.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/BluetoothSecurityManager/index.html#bt%2dsec%2dman%2eref%2eindex
Bearer independent messaging	For handling incoming messages that are intended for use by the device or for processing and not for the user to view.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/biomsg/index.html#ref%2ecpp%2ebio%2eindex
CommDb	Provides access to the underlying communications database so that applications can determine the available communications options.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/CommDb/index.html#CommDb%2etoc
HTTP Client	A standard API to use to send and receive data using HTTP.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/TFClient/index.html#ref%2ec

Library	Purpose	URL
		pp%2ehttp%2eindex
IPSec API	High-level IPSec policy and key management.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/IPSec/index.html#ref%2ecpp%2eipsec%2eindex
IrDA sockets	For data communications via infrared.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/IrDASockets/index.html#irdasockets%2etoc
MMS APIs	For sending and receiving MMS messages.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/mmsutils/index.html#ref%2emmsutils%2eindex
Posix for Symbian	A Posix API wrapper written on top of the native Symbian APIs	http://developer.symbian.com/wiki/display/pub/PIPS
Secure Sockets	For direct use of HTTPS.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/SSL/index.html#SSL%2etoc
SIM toolkit	For accessing and updating device SIM.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/SimApplicationToolkit/index.html#SATAPI_Index%2etoc
SMS APIs	For sending and receiving SMS text messages.	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/Gsmu/index.html#ref%2ecpp%2egsmu%2eindex
Sockets API	C++ sockets API similar to the popular Berkeley sockets specification.	

Table 28: Java ME Libraries

Library	Purpose	URL
Generic Connection Framework - javax.microedition.io (part of JSR-118)	Defines the Generic Connection Framework including socket, serial, push, and secure connections.	http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html
SMS APIs - javax.microedition.io.connector.sms - JSR-120	Incorporated into the MIDP specification.	same as above
Place phone call	Also part of MIDP 2.0. Use via the platformRequest function.	same as above, see also http://developers.sun.com/mobility/midp/ttips/platformRequest/index.html
Installation	This is part of the MIDlet specification or JSR-118. Describes format of the JAD and JAR files that applications must use	same as above, see also http://developers.sun.com/mobility/midp/articles/wtoolkit/
Security – javax.microedition.pki	For working with X.509 certificates.	same as above
Web Services – JSR-172	For consuming Web services.	http://java.sun.com/javame/reference/apis/jsr172/

6. BlackBerry MDS

BlackBerry MDS (Mobile Data System) is a mobile-application framework for the BlackBerry Enterprise Solution. It is efficient for creating certain types of applications. BlackBerry MDS:

1. Handles communication of application data.
2. Enables client access to internal Web services and web applications over optimized communications protocols.
3. Allows quick creation of client UIs (via the development tools) that can easily manage backend data and enterprise functionality.
4. Eliminates the need for third-party VPN software.
5. Allows a simplified programming model to support push-based applications.
6. Allows applications to be centrally controlled and managed through BES.

BlackBerry also supports two application building environments for quickly and easily creating MDS Runtime applications:

1. BlackBerry MDS Studio
2. BlackBerry Plug-in for Microsoft Visual Studio

6.1 BlackBerry MDS Studio

BlackBerry MDS Studio is a powerful IDE that is used to build MDS Runtime applications. It provides a combination of visual GUI building tools, drag-and-drop data and messaging components, and predefined user-interface objects to quickly build applications with little or no code. If you choose to add code, you write it in JavaScript.

6.2 BlackBerry Plug-in for Microsoft Visual Studio

The BlackBerry Plug-in for Microsoft Visual Studio is a component that plugs into Visual Studio 2005 IDE. This allows developers who are comfortable with Visual

Studio to continue using it instead of learning a new development environment yet have the ability to develop MDS Runtime applications. With the same features as BlackBerry MDS Studio, this can be a convenient way to start MDS application development. However, it does not let you develop MDS functionality using the .NET suite of languages such as C# and VB.NET. But it does allow you to easily integrate Web services that you develop in C# and VB.NET into your MDS application.

6.3 How MDS Works

The MDS architecture consists of three key pieces:

1. The BlackBerry MDS services
2. The BlackBerry MDS connector
3. The MDS Runtime

The BlackBerry MDS services provide the management console for centrally managing MDS client applications. The BlackBerry MDS connector allows the client device to securely and efficiently connect to back-end data sources, Web services, and Web applications. This simplifies client-side programming by eliminating the need for developers to write transport or security level code. The MDS client runtime is the component that executes MDS client applications. This runtime utilizes the same transport used by Java network-enabled applications and the BlackBerry browser to provide core services such as push, network connectivity, and secure communications. You can also use JavaScript to customize MDS client application functionality.

For more information, visit:

http://developer.att.com/rim

http://na.blackberry.com/eng/services/mobile.jsp#tab_tab_overview

7. Programming Considerations

Two fundamental considerations for mobile developers are memory management and connection management. In desktop and server environments, both of these programming aspects are rarely considerations when you develop applications due to the large amount of memory commonly available and the availability of robust and dependable network connections such as Ethernet. In the mobile environment, devices are becoming more powerful, but still have highly constrained memory. Additionally, connections cannot be assumed to be present and can range across multiple types, including EDGE, 3G and Wi-Fi.

7.1 Memory Management

Some of the important aspects of memory management include:

1. How to determine the amount of RAM and storage available
2. How to read and write to storage cards
3. General programming considerations for minimizing memory consumption

7.1.1 Determining available RAM

Although not typically required, your application may need to determine the amount of RAM available. For instance, if you know you require a certain amount of memory to load large images or databases, it is more graceful to inform the user of unavailable memory rather than crashing or exiting abruptly. Depending on the platform, there are different APIs that you can use.

Table 29: Memory Management APIs

Platform	API	URL
Windows Mobile (C/C++)	Memory Management Functions	http://msdn2.microsoft.com/en-us/library/aa909240.aspx
Symbian (C/C++)	Memory Allocation API	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/r

Platform	API	URL
		eference/cpp/MemoryAllocation/index.html#MemoryAllocation%2etoc
Palm OS Garnet (C/C++)	Memory Management API	http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/MemoryManager.html#1016260
Java ME	java.lang.runtime	http://java.sun.com/j2me/docs/pdf/cldcapi.pdf

7.1.2 Determining Available Storage Disks

It is also important for some mobile applications to store data on devices such as compact flash and SD cards. The following is a list of APIs available for the various platforms for determining available storage options.

Table 30: Storage APIs

Platform	API	URL
Window Mobile (C/C++)	Storage card API	http://msdn2.microsoft.com/en-us/library/aa916710.aspx
Symbian (C/C++)	Memory Allocation API	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/MemoryAllocation/index.html#MemoryAllocation%2etoc
Palm OS Garnet (C/C++)	Expansion Manager	http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/ExpansionManager.html#1032894

Platform	API	URL
Java ME	javax.microedition.io and javax.microedition.rms	http://java.sun.com/javame/reference/apis/jsr118/

7.1.3 Minimizing memory usage

One aspect of program design developers should be keenly aware of when designing mobile applications is how to minimize memory usage. The following list highlights some of the main techniques to do so:

1. Use local variables when possible – The compiler or runtime will allocate these on the stack and the space is then reused when the function is complete.

2. Don't use sparse arrays – applications that maintain sparse arrays for the sake of programming simplicity may incur a high memory cost.

3. Use local storage – If you are dealing with lists that may contain hundreds of thousands of items, write your algorithms so that they cache portions of the list to the expansion card when necessary

Compression – Although not typically an ideal solution for RAM, you may find that your application can benefit from using compression.

7.2 Connection Management

Due to the number of connection options and the possibility of not having a connection, you should make special considerations when handling connections in your mobile applications. The following list highlights some of the key considerations:

1. Be fault tolerant – Unlike desktop applications where you can expect to have a hard-wired network connection, you cannot expect the same with handheld devices. Your application should gracefully handle conditions where connections drop unexpectedly. Write your application so that is aware of the connection state and can react immediately when the connection is dropped. For instance, your application could have a thread

that routinely checks the signal strength and if it reaches a certain low level, that thread could trigger the data transfer thread to save its current state to minimize the impact of the dropped connection when it occurs.

2. Use the data push model if applicable – If your application only needs to perform an action when data arrives at a server, it may be beneficial to push the updates to the handheld versus having the handheld frequently poll the server for updates. Systems such as RIM BlackBerry and some mobile middleware solutions have push capability built in.

3. Compress large amounts of data – Although many communications protocols compress data, the frame size are typically too small to cause significant performance gains. Your application should compress data where possible. Systems such as the RIM BES and mobile middleware compress application data automatically.

4. Use differencing and caching – These are general schemes that you should employ to only move the parts of the data that have changed. For instance, using the Unix diff algorithm on large text files that are moved from client to server can yield significant network performance gains.

5. Data checkpoints – If your application transfers large amounts of data it is prudent to establish checkpoints during the transfer process. If the connection is lost during the transfer process, your application can then resume the data transfer at the last checkpoint instead of starting from the beginning.

The following table lists the APIs that are available on the various platforms to assist with communication management.

Table 31: Connection Management APIs

Platform	API	URL
Window Mobile (C/C++)	Connection Manager API	http://msdn2.microsoft.com/en-us/library/bb416435.aspx
Symbian (C/C++)	Connection Manager API	http://www.symbian.com/developer/techlib/v70sdocs/doc_source/reference/cpp/ConnectionManager/RConnectionClass.html

Platform	API	URL
Palm OS Garnet (C/C++)	NetLib	http://www.access-company.com/developers/documents/palmos/palmos.html
Java ME	Generic Connection Framework (javax.microedition.io)	http://java.sun.com/javame/reference/apis/jsr218/javax/microedition/io/package-summary.html

The following table lists the compression libraries available on the various platforms.

Table 32: Compression APIs

Platform	API	URL
Window Mobile (.NET)	System.IO.Compression	http://www.microsoft.com/downloads/details.aspx?FamilyID=1343d537-a62f-4a6e-9727-7791bf4cc2bd&displaylang=en
Multiple	zlib	http://www.zlib.net/
Palm OS Garnet (C/C++)	Lz77 API	http://www.access-company.com/developers/documents/palmos/palmos.html
Symbian (C/C++)	EZLIB	http://www.symbian.com/developer/techlib/v8.1adocs/doc_source/reference/reference-cpp/N102C2/CEZFileToGZipClass.html
Java ME	java.util.zip	http://java.sun.com/j2se/1.5.0/docs/api/java/util/zip/package-summary.html

8. Mobile Browsers

Recently, vendors have made significant improvements in the capabilities of their mobile browsers. For instance, Apple's iPhone contains a full-featured version of Safari that seamlessly runs most Web 2.0 applications without modification. Most of the other mobile browsers are not nearly as feature rich, but are nevertheless progressively increasing in functionality. The following table lists the most popular mobile browsers as well as their capabilities.

Table 33: Mobile Browser Summary

Browser	Devices	Ajax	DHTML	Flash
Pocket IE	WM	Yes	Yes	Yes
Opera Mobile 8.65, 9.5	WM, Symbian	Yes	Yes	Yes
Opera Mini 4	WM, Symbian	Mixed	Yes	No
Web Browser for S60	Symbian	Yes	Yes	Yes
Blazer 2, 3, 4	Palm	No	No	No
BlackBerry Browser	BlackBerry	No	No	No

When developing web applications which target mobile browsers, here are some things to keep in mind.

1. Even though many of the browsers support client-side scripting, they do not necessarily support it fully. Many callbacks and events do not happen and many parts of the Document Object Model (DOM) are often not available.

2. Browsers have improved significantly in being able to render pages so that they look approximately the same on the device as on the desktop. However, applications that are specifically designed for a small form factor will still have better usability.

3. WebKit is the basis for many browsers including Safari for the iPhone, Web Browser for S60, and the Android reference browser. Issues that you encounter with any of these browsers may be solved by searching the WebKit documentation and forums.

8.1 XHTML Mobile Profile

The XHTML Mobile Profile is the HTML specification for mobile phones and other devices with limited inputs and screen sizes. It is defined by the Open Mobile Alliance and is defined as a strict subset of the XHTML Basic specification. It contains additional modules, elements, and attributes that allow you to create richer interfaces for mobile devices. When declaring support for XHTML Mobile Profile, a conforming user agent MUST use an Accept header with a value of "application/xhtml+xml".

8.2 Changing the Browser Location

Mobile Web applications often require that you set the mobile browser location to a specific internal website that hosts the enterprise web application. Each of the mobile browsers has a different method to either change the homepage or to navigate the default browser.

1. BlackBerry – On BlackBerry devices you can use the 'net.rim.blackberry.api.browser' API to navigate the browser to a particular page. Using this method, you could create a Java ME launch application that navigates the browser to the start page of your web application instead of actually changing the home page of the browser.

2. Pocket IE – You can modify the home page by setting the value of the 'home' key in the registry at

   HKLM\SOFTWARE\Microsoft\Internet Explorer\AboutURLs

   to the URL you wish to use as the new home page. Additionally you can open the default browser on a Windows Mobile device by calling ShellExecuteEx with the lpFile parameter set to the desired URL.

8.3 Flash Lite

In addition to basic web applications, you also have the ability to use a version of the Macromedia Flash player designed specifically for mobile and resource-constrained devices: Flash Lite. The main benefit you receive by building a Flash Lite application versus a Web application is that your application will look the way you expect on all supported devices. The current drawback with Flash Lite is that it is only supported on Symbian and BREW devices. However, since Flash Player 7 is available for Windows Mobile platforms, it is feasible to create a version of your application for both sets of platforms. The key benefits to using Flash Lite are:

1. Consistent user interface across all supported platforms.
2. Flash Video (FLV) support for video.
3. Supports most Flash 8 content.
4. Integrated authoring environment with Adobe Dreamweaver CS3.

Some of the drawbacks to using Flash Lite are:

1. May not have the Flash 8 compatible features that you need.
2. Not nearly as powerful as a Java ME or native application.
3. Not an option for BlackBerry and Palm.
4. Cannot access unique device features.

9. Middleware Applications

A wide-variety of mobile middleware applications are available to help you quickly create enterprise applications. Most of these middleware applications are designed to assist with quickly creating mobile applications that can access backend databases and Web services. The main advantages of the middleware approach are:

1. Rapid Application Development – Developers can create applications faster and more easily than with C++ and Java

2. Insulates Developer – Most middleware architectures handle connection management, data caching and database shadowing so that developers can concentrate purely on the business logic of the application

3. Multi-Device Support – Many middleware platforms support multiple mobile device platforms from the same code base

4. Simplifies User Interface Creation – Middleware tools typically create the UI and provide tools to modify it based on the underlying data model. Developers spend less time creating and maintaining the UI for multiple mobile form factors

Although there are advantages to middleware architectures, there are also several drawbacks:

1. Lack of Flexibility – If the functionality your application requires is not built into the middleware platform, it will be difficult or impossible to integrate it into your application.

2. Device Support – The middleware platform must support all of the devices that you intend to allow your application to run on.

3. Custom Backend Support Considerations – In some cases, it may be difficult for the middleware server to support custom backend databases or CRM systems. Depending on application complexity, you may have to write custom backend code to expose the data store to the middleware server.

For more middleware information, visit http://developer.att.com/middleware.

10. Debugging

Mobile application debugging is another area different from desktop applications. Some of the key differences are:

1. Mobile applications require debugging in an emulator.
2. Remote debugging is frequently useful.
3. Logging debugging messages on the device is more complicated, but can be useful for field situations.

10.1 Emulator Debugging

The first and simplest phase of application debugging occurs in the emulator. The emulator is simply an application that runs on your desktop and behaves like the mobile device. It is extremely helpful since it allows you to run your application and quickly single-step through any part of your code. Using this method, you can look at variable values, set breakpoints and update memory and variable values. All of the IDEs include emulator debugging support. Additionally, you can usually get specific emulator images directly from the mobile device manufacturer.

10.2 Remote Debugging

Sometimes it is difficult or impossible for you to debug a problem within the emulator. This usually occurs when you are using functionality that is specific to a device or cannot be emulated. This could include items such as GPS or wireless-data connections. In this case, you can use remote debugging tools to find and fix the problem. This is where a debugger operating on your development PC connects to your application, either instrumented or via a debug connector on the device, allowing you to perform most of the debugging functions such as single-stepping, value updates, and variable watches that you can perform in the emulator. The connection to the device can be over links such as USB or Bluetooth.

With BlackBerry, you just connect your BlackBerry via USB and in the JDE go to Debug > Attach To > Handheld > Your PIN.

The table below lists the various APIs and documentation available for debugging.

Table 34: Debugging APIs

Platform	API	URL
Window Mobile	Remote Tools	http://msdn2.microsoft.com/en-us/library/ms894603.aspx
	Debugging API	http://msdn2.microsoft.com/en-us/library/aa910695.aspx
Palm OS Garnet (C/C++)	Debug Manager	http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/DebugMgr.html
Symbian (C/C++)	On-device debugging overview	http://www.forum.nokia.com/info/sw.nokia.com/id/1bdbfff2-c190-4558-9c17-facc9e247be4/Carbide_cpp_On_Device_Debugging_v1_1_en.pdf.html

10.3 Debug Logging

Finally, it can work well to log debug messages for diagnosing problems. This method may also be useful if you are trying to isolate a problem that is occurring on a customer device that you don't have physical access to. In this case, one typical approach is to:

1. Log the debug messages to a file or record store.
2. Create a Web service (or comparable approach) that the device can use to upload messages.
3. Update the client application to provide the user with an interface to submit debug logs.
4. Alternatively, send log information via e-mail.

Using the techniques above will allow you to not only isolate and fix issues that you know about today, but also in the future when you no longer have physical access to the devices where the issue is occurring.