

# AT&T Developer Program

## Mobile Application Development Best Practices

### White Paper

Document Number **10001**  
Revision **1.0**  
Revision Date **9/23/2007**

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.

© 2007 AT&T Knowledge Ventures  
All rights reserved.

AT&T, AT&T logo, Cingular and Cingular logos are trademarks of AT&T Knowledge Ventures and/or AT&T affiliated companies.

## Revision History

All marks, trademarks, and product names used in this document are the property of their respective owners.

<b>Date</b>	<b>Revision</b>	<b>Description</b>
9/22/2007	1.0	First release

# Table of Contents

1. Introduction .....	1
1.1 Audience .....	1
1.2 Contact Information .....	1
1.3 Resources .....	2
1.4 Terms and Acronyms.....	2
2. Scope .....	4
3. Unique Aspects of Mobile Applications .....	5
4. Formulate a Wireless Strategy.....	6
4.1 Gather Relevant Documentation .....	6
4.2 Identify and Interview Key Stakeholders and Representative Users.....	6
4.3 Research Industry and Competitive Trends.....	6
4.4 Conduct Strategy Sessions .....	7
4.5 Identify Key Mobile Processes and Value Drivers.....	7
4.6 Examine Impact on Key Business Processes and Reengineer the Processes as Necessary ....	8
5. Evaluate Technology Requirements and Available Approaches .....	9
6. Fit the Mobile Solution to the Environment.....	11
7. Define a Platform Policy .....	15
8. Remember the Mobile Device Is Not the Desktop .....	16
9. Analyze Coverage .....	18
10. Manage Mobile Devices .....	19
11. Design for Security .....	20
12. Conclusion .....	22
13. Appendix A: Mobile Application Development Steps .....	23
13.1 Prioritizing Mobilization Opportunities.....	23
13.2 Mobile Application Development Steps .....	24
14. Appendix B Mobile Application Development Reference .....	28
14.1 Mobile Application Architectures .....	28
14.2 Mobile Application Development Technologies.....	33
14.3 Mobile Application Development Tools.....	37

# Table of Contents

14.4	Capability Summary of Mobile Platforms .....	38
14.5	Throughput and Latency .....	39
14.6	Battery and Power Management .....	40
14.7	Input, Text, Screens, Usability .....	42
14.8	Memory Management .....	43
14.9	Network .....	45
14.10	Connection Management .....	46
14.11	Security .....	48
14.12	Push vs. Pull .....	50
14.13	Telephony, SMS, SIM Interfaces .....	53
14.14	Communication Cost Management .....	54
14.15	Managing Amount of Data Communicated .....	55
14.16	Recovery and Diagnostics .....	56

## Figures

Figure 1:	Development/Support Cost vs. Usability/Sophistication .....	10
Figure 2:	Middleware Architecture .....	12
Figure 3:	Project Prioritization Matrix .....	24
Figure 4:	Web 1.0 Example .....	35
Figure 5:	Web 2.0 Version of the Same Application .....	36
Figure 6:	Handheld Device Communication Architecture .....	47
Figure 7:	Security on an End-to-End Basis .....	50
Figure 8:	Pull Architecture .....	51
Figure 9:	Push Architecture .....	51

## Tables

Table 1:	Terms and Acronyms .....	2
Table 2:	Strengths and Weaknesses of Mobile Computing Solutions .....	13
Table 3:	Pros and Cons of Different Development Approaches .....	31
Table 4:	Mobile Platform Development Matrix .....	37
Table 5:	Different Capabilities of Mobile Devices .....	38
Table 6:	Wireless Network Throughput Speeds .....	39
Table 7:	Wireless vs. Wireline Networking .....	46
Table 8:	Some Common APIs for Connection Management .....	47
Table 9:	SMS and SIM APIs for Handheld Platforms .....	54
Table 10:	Compression APIs for Handheld Platforms .....	55

## 1. Introduction

With the increases in wireless network bandwidth and demand from mobile users, mobile data usage is exploding. Companies are under mounting pressure to take advantage of the bandwidth. For example, sales managers want customer relationship information at their fingertips, delivery personnel want up-to-date location information, and fleet managers want to maximize the productivity of their workers.

What about packaged applications? Several good options are available across all verticals, with new applications coming out every day. Many of these applications allow some customization. However, as mobile application deployments mature, enterprises are finding the need to do extensive customization. Facing make/buy decisions, more enterprises are looking to develop custom applications to meet their precise needs. In some cases, solution integrators can offer a viable alternative.

Many developers began creating applications for servers or personal computers and then moved to mobile devices. The extensibility of tools and common programming environments have made this transition fairly straightforward, although developers do have to address issues specific to mobile computing and wireless networking. Developers are often the first people in a company to carry out such mobilization efforts, and the enterprise may not have built up a set of best practices. Many mistakes have been repeated, so AT&T presents this white paper as a guide.

### 1.1 Audience

The target audience of this white paper is IT developers, architects, and managers looking for common best practices for their mobile development efforts. In particular, the paper is geared toward experienced developers 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>

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

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

APN Information: <http://developer.att.com/apn>

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
3GPP	Third Generation Partnership Project
Ajax	Asynchronous JavaScript And XML
API	Application Programming Interface
APN	Access Point Name
CPU	Central Processing Unit
CRM	customer relationship management
CVS	Concurrent Versioning System
DHCP	Dynamic Host Configuration Protocol
DHTML	Dynamic Hypertext Markup Language
DMZ	Demilitarized Zone
EAI	Enterprise Application Integration
EDGE	Enhanced Data Rates for GSM Evolution
GUI	Graphical User Interface
GSM	Global System for Mobile communications
HSDPA	High-Speed Downlink Packet Access

Term or Acronym	Definition
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol over Secure Sockets Layer
I/O	Input/Output
IDE	Integrated Development Environment
IP	Internet Protocol
IPsec	Internet Protocol Security
ISP	Internet Service Provider
Java ME	Java Platform Micro Edition
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
Kbps	Kilobits Per Second
LAN	Local Area Network
Mbps	Megabits Per Second
MIDP	Mobile Information Device Profile
PDF	Portable Document Format
PDP	Packet Data Protocol
PNG	Portable Network Graphics
RAM	Random Access Memory
SDK	Software Development Kit
SIM	Subscriber Identity Module
SMPP	Short Message Peer-to-Peer Protocol
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SSL	Secure Sockets Layer
SVG	Scalable Vector Graphics
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
VM	Virtual Machine
VPN	Virtual Private Network
WAP	Wireless Application Protocol
WAV	Waveform Audio Format
XML	eXtensible Markup Language

## 2. Scope

The potential scope of a white paper on mobile application development is huge. It could address many topics, including the choice of platforms, mobile operating systems, mobile middleware, and development tools. The focus in the main body of this paper is guiding principles that can ease development and speed the development process. These include best practices for formulating a wireless strategy and tips on solution design, development, deployment, and maintenance. The paper also covers analysis and optimization of wireless solutions, which involve a variety of factors and issues related to design, development, implementation, and deployment.

Appendix A provides an overview of the steps involved in mobile application development. Appendix B provides practical information on specific aspects of mobile application development, including mobile computing architectures, developer tools, bandwidth and latency details, power management, memory management, push versus pull, user interfaces, networking considerations, and security. The goal is to give readers an initial context for understanding the unique aspects of mobile application development. AT&T has extensive material available for its developers covering many of the topics raised in this white paper in greater detail. Beyond that, extensive information is available from other parties, such as mobile operating system vendors.

### 3. Unique Aspects of Mobile Applications

Mobile applications are not just desktop applications reformatted for a small display. They are fundamentally different for many reasons. First, the ability to communicate from anywhere essentially changes how users interact with an application. For example, a batch process that previously required a user to upload information at the end of the day is now a dynamic, interactive process that occurs throughout the day. Second, the ideal user interface for a small screen and a small (if any) keyboard differs significantly from desktop/laptop systems. Third, the types of communications channels are different. Mobile devices incorporate voice capability, messaging capability (that is, Short Message Service [SMS]), location information, and now video. The best mobile applications integrate these capabilities to optimize how users interact with data. Finally, the nature of wireless networks is different than that of wireline networks. Though today's wireless networks offer broadband data capabilities, throughput can vary based on signal quality, and a network connection is not always available, particularly if users are mobile.

## **4. Formulate a Wireless Strategy**

Prior to embarking on specific initiatives for the development and deployment of mobility solutions, AT&T recommends defining a formal process for setting an overall mobility strategy. This strategy formulation process is similar to the strategic planning process used in other disciplines within the organization. The recommended process includes the following steps:

### **4.1 Gather Relevant Documentation**

First, gather and review all existing strategy documents that may be relevant to the organization's objectives with respect to wireless mobility and mobile business processes in general. These documents will help ensure the alignment of any mobility initiatives with business objectives and the existing enterprise architecture.

### **4.2 Identify and Interview Key Stakeholders and Representative Users**

Prior to scheduling actual strategy formulation sessions, develop a list of questions that will begin to bring some direction to the strategy formulation process. Identify and interview key stakeholders, including any mobile device end users. The interviews should serve two purposes. First, they should help uncover any existing mobility projects or strategy elements and establish the foundation knowledge of the existing enterprise architecture with which the new mobile architecture must integrate. Second, interviews are an opportunity to identify, understand, and document processes that may benefit from more data in more places. Key stakeholders can be helpful in identifying important mobile processes and providing a preliminary understanding of those processes. But representative users--the people who actually execute the processes on a daily basis--generally will provide a much more accurate and detailed picture of the processes they perform, how they perform them, and how the processes could be improved if mobile data were available.

### **4.3 Research Industry and Competitive Trends**

Before the session begins, it is important to research what others in the same and related industries have done. Understanding what competitors and others in

similar industries are doing with mobility provides an excellent starting point. This information can be gathered in a number of ways. First, visit the Web sites of key competitors and of companies in related industries to see if they list strategic imperatives and, if they do, whether mobility is mentioned as a way to address these imperatives. One easy way to gather this information is to request it from your company's wireless carrier, as that provider may have done projects in similar industries and subsequently be able to provide additional insight. AT&T has a collection of case studies on previous work done in many industries.

#### **4.4 Conduct Strategy Sessions**

During strategy sessions, address the following topics:

- Identify key mobile processes and value drivers.
- Examine the impact on key business processes and reengineer the processes as necessary.
- Evaluate technology requirements and opportunities.
- Prioritize opportunities for wireless enablement.
- Develop high-level project charters for initial projects.
- Establish key principles and next steps for the go-forward plan.

#### **4.5 Identify Key Mobile Processes and Value Drivers**

This is the most important, and often the most difficult, part of a strategy session. Resist the impulse to think in terms of mobilizing desktop applications; rather, think of improving existing or enabling new processes in mobile environments. Identify and catalog activities employees perform away from their offices. Include processes that are not currently automated, in addition to possible processes that are not currently performed because the required data is unavailable where and when it is needed. Create "as is" process maps for each process identified. Use the process maps to identify possible points of leverage, where better, more timely data availability could save time and money, drive revenue, or improve customer satisfaction. To develop the best possible process maps, it may be necessary to do a "day in the life" exercise. These exercises involve following users around on a typical workday and documenting what processes they perform and how they perform them. Do not forget to ask the most important question, "Why do you do it that way?"

## **4.6 Examine Impact on Key Business Processes and Reengineer the Processes as Necessary**

Mobile data has the potential to dramatically change the way employees work. Starting with the leverage points identified in the previous step, determine how the process can be streamlined, simplified, or improved to maximize the value generated. Do not be overly influenced by existing processes, especially if those processes are currently performed in an environment different from the target mobile environment.

The process may often need to be created from the ground up, leveraging new capabilities that are made available by the mobilization of data. This is particularly true if the process is performed manually or has historically been desk bound. Creative thinking is vital to driving value from data mobilization, so keep an open mind when examining alternative ways to perform a task. Create new, proposed process maps that identify areas of improvement over the previous process, the strategic imperatives they serve, and the value they create.

## 5. Evaluate Technology Requirements and Available Approaches

When doing mobile development, it is important to select the technology that best serves the process or processes identified through the procedures discussed above. At this point, the technical requirement can be high level and simply establish a general direction and broad technology choices. The goal is not to produce a technical solution to address the process changes; rather, it is to provide a rough assessment of the level of effort that might be involved in data mobilization. Because a variety of application approaches are available, you will need to decide between a solution built in house, a commercial solution geared to your company's specific application needs, or a more flexible mobile middleware development platform.

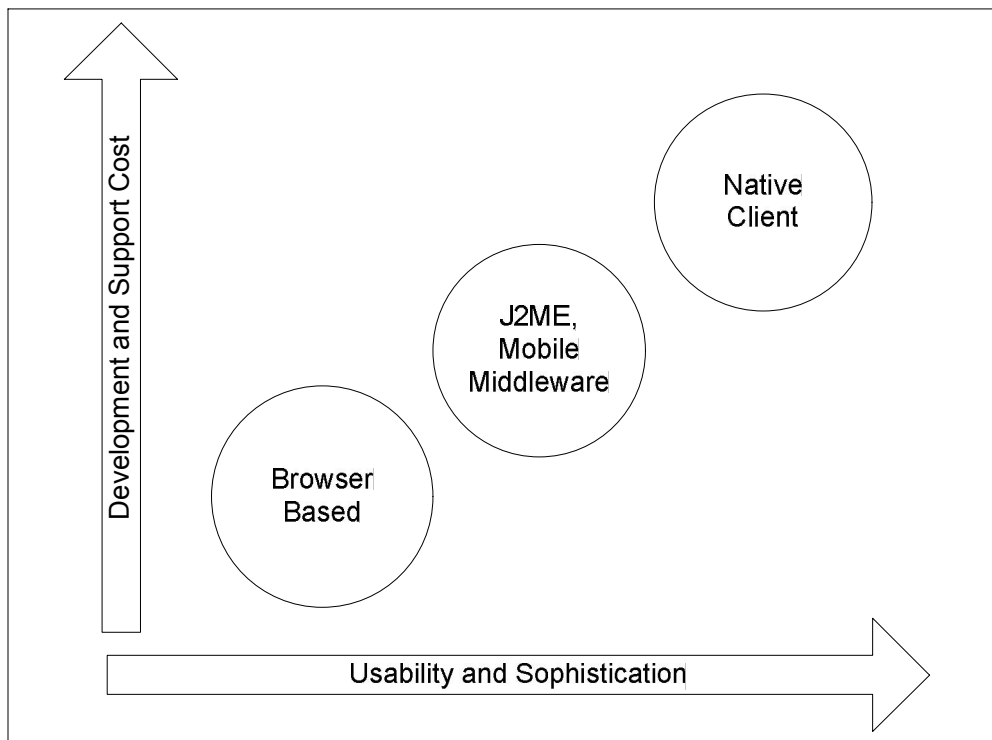
The high-level decisions you will need to make should include the following considerations:

- **Browser-based architecture?** A browser-based approach requires relatively simple development and makes your application available on a wide variety of handheld devices. Among the disadvantages are slower operation and the need to always have a network connection.
- **Rich versus thin client?** More powerful smartphones can support a sophisticated user interface with significant local data processing, but less capable devices may benefit if most processing is done at the server.
- **Store-and-forward architecture?** These applications (for example, wireless e-mail) queue transactions based on the availability of connections, so operation is possible even when the device is out of the coverage area. This is a more powerful, but more complex approach.
- **Push versus pull?** A push application on a server automatically sends new information to a mobile client (that is, many wireless e-mail systems), whereas a pull application on a client queries the server for new information.
- **Custom or native development?** Smartphone systems support applications that are either developed in languages such as C++ and Java or use mobile middleware platforms that can provide higher level

interfaces. The learning curve associated with custom development can be significant.

The following figure shows how development and support costs increase with different architectural approaches, as do usability and sophistication.

**Figure 1: Development/Support Cost vs. Usability/Sophistication**



The Appendices offer more information about the pros and cons of these approaches and the available tools.

## 6. Fit the Mobile Solution to the Environment

Various approaches are available for integrating mobile solutions with existing enterprise applications. Which works best depends on your company's particular circumstances and objectives.

A number of vendors have developed targeted applications for specific work functions, including field service, transportation, medical care, inventory control, and logistics. These are referred to as vertical-market applications. Whether or not they include management and security features depends on the vendor and the application.

Wireless e-mail (for example, RIM BlackBerry, Motorola GoodLink) and synchronization systems (for example, Nokia Intellisync) synchronize not only e-mail but also calendars and contact information. Moreover, some of these systems (for example, RIM Mobile Data Service) provide general-purpose access to enterprise information, and many also have management and security features.

With the growth in mobile and wireless computing, most enterprise application vendors now have mobility extensions for their products. For example, Microsoft supports wireless e-mail with Microsoft Exchange and mobile instant messaging with Office Communications Server. A recent *Network Computing* article surveyed major application vendors and presented details on handheld device support from IBM, Oracle, Salesforce.com, SAP, and Sybase.<sup>1</sup> The bottom line is that the first place to look for mobility support is from vendors with which your organization may already be working. Note that the mobility extensions for enterprise applications often employ middleware architectures. However, vendors in this category typically have mobility solutions that interoperate only with their own products.

Examples of enterprise application architectures that can benefit from mobility extensions include those in which the same database, application server, or enterprise application integration (EAI) suite is used for most of the applications deployed within the enterprise. For example, if your company's enterprise applications are deployed on IBM Websphere and MQseries, you should consider using the IBM mobility components Websphere Everywhere Access (WEA) and WebSphere Everywhere Connection Manager (WECM).

---

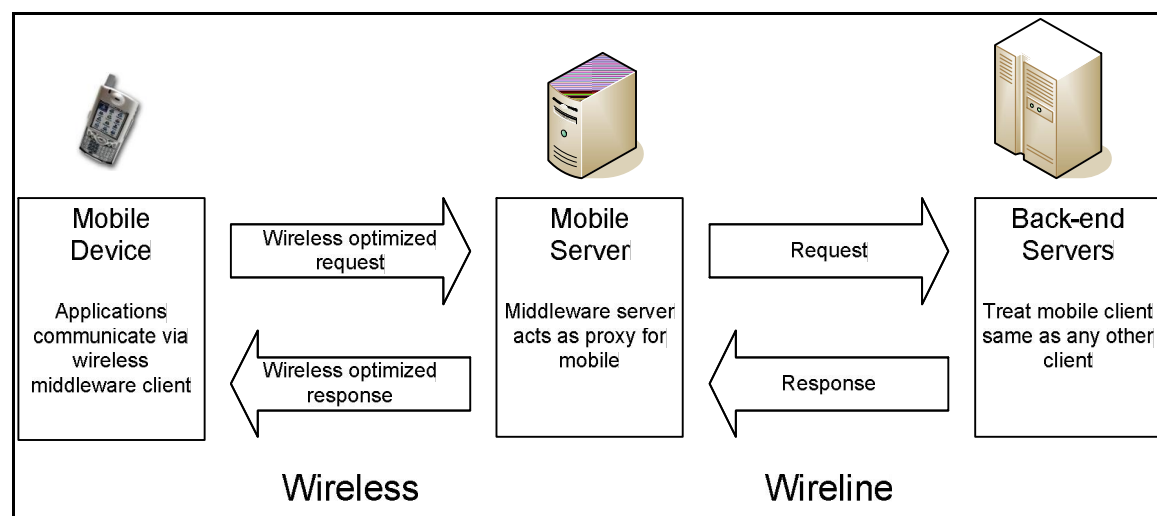
<sup>1</sup> Rysavy Research article: "Reach Me if You Can," <http://www.rysavvy.com/papers.html>, May 2007

Finally, general-purpose mobile/wireless middleware products are designed to provide mobile devices with access to a wide variety of back-end services potentially spanning multiple vendors. Wireless middleware solutions sometimes provide their own Application Programming Interfaces (APIs) with which you can develop mobile applications. Alternatively, APIs provide non-programmatic approaches to support mobile data. These types of products have other names. For instance, Gartner Inc. refers to these products as Multichannel Access Gateways. Examples of companies in this category include Antenna, Vetro, and Dexterra. Although the device-side runtime is necessarily unique to the device type and possibly even the specific device, the stronger platforms support multiple mobile devices through a single client application. The middleware handles all of the device-specific attributes. If a new device is introduced with a different form factor or different capabilities, an update from the platform vendor will typically allow existing code to run on the new device.

Keep in mind, however, that middleware platforms are complex. While ultimately simplifying development, these platforms have their own learning curve. You will also have to consider client and server licensing fees.

The following figure shows how typical middleware architectures include mobile servers that act as proxies for mobile devices and employ wireless-optimized protocols over the radio link.

**Figure 2: Middleware Architecture**



If none of the “off-the-shelf” approaches described above address your company’s needs, you can develop a custom application. For many enterprises, this will be the most difficult alternative; however, it can provide exactly the functions your company needs. The Appendices provide extensive guidelines if you choose this approach.

The following table summarizes the options discussed above.

**Table 2: Strengths and Weaknesses of Mobile Computing Solutions**

Type of Mobile Solution	Strengths	Weaknesses
<b>Vertical Market Solutions</b>	<ul style="list-style-type: none"> <li>• Can provide the most comprehensive features for targeted work functions</li> </ul>	<ul style="list-style-type: none"> <li>• Not generally extensible to other enterprise data</li> </ul>
<b>Wireless E-mail and Synchronization Products</b>	<ul style="list-style-type: none"> <li>• Relatively easy to integrate</li> <li>• Function well for core applications</li> <li>• Some extensions available for other enterprise data</li> <li>• Some support a wide range of mobile devices</li> <li>• Management and security features generally included</li> </ul>	<ul style="list-style-type: none"> <li>• May not provide access to all application data of interest</li> </ul>
<b>Mobility Extension from Existing Application Vendor</b>	<ul style="list-style-type: none"> <li>• Working with just one vendor</li> <li>• Often the simplest approach</li> <li>• Management and security features generally included</li> </ul>	<ul style="list-style-type: none"> <li>• Solutions usually only support the vendor’s applications</li> <li>• Range of mobile devices supported may not be as great as other approaches</li> <li>• Device may not be able to support</li> </ul>

Type of Mobile Solution	Strengths	Weaknesses
		multiple vendor solutions simultaneously
<b>General Purpose Mobile/Wireless Middleware Solutions</b>	<ul style="list-style-type: none"><li>• Greatest range of applications supported</li><li>• Greatest range of mobile devices supported</li><li>• Sometimes provide APIs for custom applications</li></ul>	<ul style="list-style-type: none"><li>• Systems are complex and have a learning curve</li><li>• Integration may be more complex than other approaches</li></ul>

## 7. Define a Platform Policy

In the mobile space, enterprises and users can choose from a variety of platforms and devices. Devices come in many form factors, with vastly different functionality. The range of handheld operating systems includes proprietary systems, Garnet OS, iPhone, Mobile Linux, Symbian, and Windows Mobile. Although this diversity enables solutions for almost any user need, it can easily lead to interoperability issues. Your company should consider restricting the number of platforms it supports, because each has its own application support requirements, in addition to management and security considerations.

As a start, you need to choose between deploying a mobile application on existing devices or purchasing new types of devices. Considerations include whether the device has the processing power and memory to support your desired applications, and whether security and management systems are available for the device. Management is particularly important as the number of supported users increases. For example, tools that automatically update software over the air can be particularly important in large deployments.

Consider, also, the lifecycle of a device. If your devices need to be available for a long-term deployment and new devices need to be added, you should avoid those devices that are close to the end of their lifecycle. Similar considerations apply to mobile operating systems.

## 8. Remember the Mobile Device Is Not the Desktop

This may be an obvious point, but it is a very common mistake. Some tools make it possible to create an application for the desktop and port it to a mobile device. But that does not mean the resulting application will be a good wireless application. Rather, treating a mobile device the same as a desktop will almost guarantee that the wireless project will fail.

A number of factors make mobile applications fundamentally different than desktop applications.

1. The device has a more constrained user interface. Often, the user interacts with the device using just one hand.
2. The wireless network connection generally operates slower than a local area network (LAN) connection, and a connection is not always available. Sometimes, connections can be lost in the middle of a transaction.
3. Users may work differently when outside the office compared to inside the office.

During application design, you need to accommodate how and when the application will use the wireless network, the type of device or devices the application will run on, and security requirements. The Appendices present more information on how to address these specific items.

Most of the application design tasks involve starting with functional design and high-level technology choices and, based on your choices, adding enough extra information to develop a detailed specification and project plan.

How should you begin? Project definitions can outline the features and functions your mobile application must implement. This includes, at a minimum, listing the screens, the fields within the screens, and the application flow diagrams. Even if an existing wireless application is in place and your planned application is a replacement, AT&T recommends a thorough design and development cycle that includes the following steps:

- Business process and workflow modeling
- Requirements definition
- Functional specification

- Systems design
- Application design
- Application development
- Test plan development
- Test plan execution

Although these steps are common in all types of application development, there are considerations specific to mobile platforms. These steps and the associated mobile considerations are discussed in the “Mobile Application Development Steps” section of Appendix A.

This white paper presents specific recommendations for debugging, including recovery and diagnostic mechanisms, in the “Recovery and Diagnostics” section of Appendix B.

## 9. Analyze Coverage

AT&T provides widespread national coverage with its Global System for Mobile Communications/Enhanced Data Rates for GSM Evolution (GSM/EDGE) and Third Generation (3G) networks. Many companies and users take advantage of the network without strictly considering exactly where coverage is available.

However, depending on your company's application and work process requirements, you may need to verify that your users have coverage where it is needed, especially if they work in known or predictable areas. Such a coverage analysis can be theoretical or empirical.

Theoretical coverage analysis provides a coverage map based on theoretical or mathematically generated coverage data. These maps are often called propagation maps. Many wireless service providers produce these types of coverage maps in varying levels of detail; however, maps generally only cover larger geographic areas that show coverage with respect to populated areas and highways and not necessarily to the street level.

Empirical coverage analysis provides detailed coverage maps based on an actual coverage field test within a desired service area that uses actual received signal strength information from a wireless device. Coverage information is captured by driving the exact routes of typical wireless system users with wireless devices and GPS and by plotting the combined location and signal strength information on a map. This type of analysis requires some level of expertise. You can also do a less formal analysis by monitoring signal strength and running test transactions from desired coverage areas.

## 10. Manage Mobile Devices

As the number of mobile devices your company deploys increases, managing these devices becomes ever more important. When enterprises first introduced mobile phones, they typically did not have policies to provision or manage these devices, and employees often would purchase them on their own. With access to enterprise data, however, mobile devices pose a much greater security risk than the voice-only devices of the past. At a minimum, AT&T recommends that your company protect against lost or stolen devices by being able to remotely wipe data and disable the affected device.

Mobile device management is an indispensable part of any mobile solution deployment of more than a few devices. In the absence of a mobile device management tool, device usage cannot be monitored and device logs cannot be examined. Mobile device management serves several purposes.

- It provides the tools needed to enforce security measures.
- It helps support lower cost and better service from the enterprise's helpdesk.
- It allows software to be updated over the air.
- It ensures that the device has the correct version of software.
- It provides the IT team a powerful diagnostic capability in the event that a problem with the device, application, or network arises.

## 11. Design for Security

Enterprise security has generated a lot of expertise, but the unique elements of wireless security are less well known. Security actually delays or kills some mobile deployments; however, that need not be the case. Multiple solutions and approaches are available, including passwords, methods to disable lost devices, virus protection, and extensive security features within the wireless network, such as radio link encryption.

One security issue is that some mobile application platforms and some mobile point solutions provide their own application layer security. Application layer security typically requires you to locate the mobile application server in the perimeter network (for example, the demilitarized zone [DMZ]) or open additional ports on the firewall. Some enterprises prefer not to use these configurations. If your company chooses this path, you can use a Virtual Private Network (VPN) to provide end-to-end security. If you use a VPN, you should keep the following points in mind:

- VPNs are sensitive to connection availability, so if you employ these networks in mobile environments, your users may occasionally need to restart VPN sessions.
- VPNs designed for wireless networks (mobile VPNs) can tolerate connectivity loss, and they offer optimizations that can actually increase your application's performance.
- Client VPN software, particularly for Internet Protocol Security (IPsec) VPNs, may not always be available for the device you choose.
- Most mobile browsers support Secure Sockets Layer (SSL), meaning that SSL VPNs can support a large number of handheld devices.

AT&T also offers a number of security features and options with its data services that you should understand and leverage in your security architecture. Additional security considerations are discussed in "Security" section of Appendix B.

Although not many large-scaled mobile security breaches have been publicized, that does not mean mobile security is not an issue. In a recent survey<sup>2</sup>, 83 percent of wireless carriers had seen malware on mobile devices, including 50

---

<sup>2</sup> Informa Telecoms and Media, February 2007

percent that had seen it in the previous three months. In addition, there have been several instances where mobile devices were lost and information was compromised. AT&T will do its part to support mobile security, but individual companies have to take their own actions and assume responsibility for their mobile devices, the data these devices contain, and the ability of these devices to access enterprise data.

## 12. Conclusion

Now is the time for enterprises to mobilize their business processes. Wireless development and deployment are similar to other software development efforts in some respects. But there are many specific considerations you need to incorporate into your mobile application planning and deployment. This white paper has described AT&T's view of the best practices for working through the solution design, development, deployment, and maintenance phases, in addition to providing information about wireless solution diagnosis, analysis, and optimization.

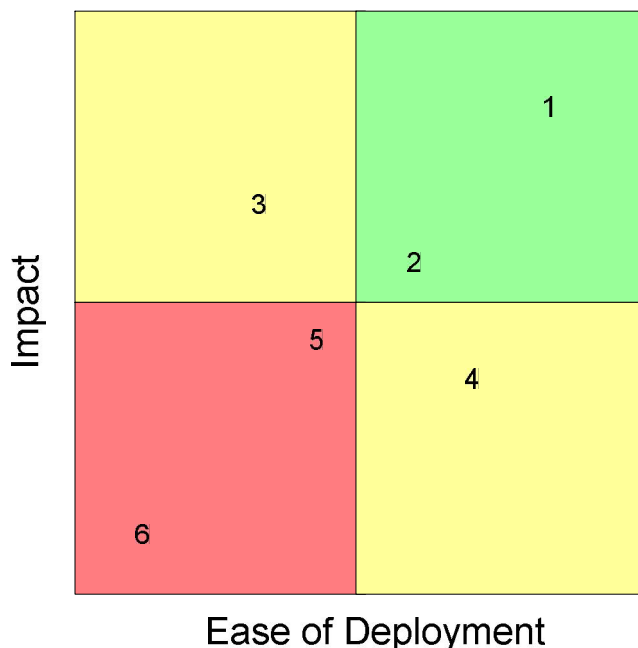
## 13. Appendix A: Mobile Application Development Steps

### 13.1 Prioritizing Mobilization Opportunities

It is important that your company's initial mobile projects be successful, so you must take the time to separate the most promising opportunities for process mobilization. A good way to do this is plotting each of the initiatives on a magic quadrant graph. The figure below depicts a representative graph on which six hypothetical projects have been plotted. Impact is on the Y axis and captures the net impact on your business; it may be computed as benefits minus costs. Higher positioning on the Y axis means higher impact. This need not be an exact estimate, but it should be directionally accurate. Ease of deployment is plotted on the X axis. It represents the amount of effort necessary to develop, deploy, and maintain your mobile solution. Efforts that are farther to the left are more difficult or complex, while efforts that are farther to the right are less difficult or complex. Using these representations, it is possible to examine your mobilization options and prioritize your mobile initiatives.

Continuing the example in which six potential projects are being evaluated, Project 1 is low-hanging fruit--high impact and easy to implement. Project 2 is a bit lower impact and a bit more difficult to implement, so it should be prioritized lower than Project 1. Project 3 has a high impact but is more difficult to implement, so it should be prioritized lower than the first two projects. Conversely, Project 4 is less meaningful than the first three projects, but it is relatively easy to implement. The relative prioritization of Projects 3 and 4 depends on whether the enterprise favors high-impact projects or easy-to-implement projects. Project 5 is on the cusp, not quite meaningful enough and not quite easy enough to be given serious consideration in the first round of application selections. Project 6 would have little impact, is very difficult to implement, and should not be undertaken.

**Figure 3: Project Prioritization Matrix**



The final step is to reconcile the selected project with your company’s strategic imperatives to assure that your project positively impacts the most important objectives. Map out the remaining steps required to complete your final prioritization and selection of mobility projects, and then create a project plan detailing how and when these tasks will be addressed. It is important at this point to establish guiding principles and set priorities within your projects. Decisions should be made based on your company’s preference for ease of deployment versus importance. Priorities should also be fine-tuned based on how each of your projects aligns with the strategic imperatives of your enterprise.

## 13.2 Mobile Application Development Steps

This section describes a formalized process for mobile application development. Note that many of the recommended steps for mobile development are no different than those for desktop or server platform development.

### Business Process and Workflow Modeling

Business process and workflow modeling helps you identify those areas in which wireless communications can enhance your company’s efficiency and

effectiveness. This analysis provides valuable insight as to where and how a wireless data solution can enhance and streamline your current business practices or how an existing mobile solution can be enhanced to accommodate process changes. Information derived from this analysis can serve to monitor workflow, gauge the efficiency of your current applications, identify problem areas, identify areas of potential process improvement, and help improve support to your field operations.

## **Requirements Definition**

The wireless solution design and development process begins with the creation of documentation that fully defines the objective (what the system is expected to do and how it will be used) and the requirements (the software, hardware, implementation, training, and user documentation needed to accomplish the objective). These documents specify the functional and technical requirements of your product. The requirements definition describes not only what the product will do from the users' perspective, but also how the product will interface and work with your existing systems.

For wireless enablement, you should pay special attention to use cases and operational environments. This step is commonly skipped. In wireless, there are considerations distinct from wired applications. These include where users will access the application and what else they will be doing, in addition to the physical environment in which the application will be used (for example, low/high temperature environments, dirty or corrosive conditions, standing/walking/driving situations, low/no signal conditions). The requirements definition also addresses the availability, robustness, and support considerations of your proposed system. Navigation is completely different in wireless applications, and it should be specifically designed to your target device and platform.

## **Functional Specification**

The functional specification enumerates all the components and interfaces that make up your wireless data communication solution. This specification will include your system's functionality, in addition to its interaction and interoperability with the other systems it accesses. During functional specification, you should work closely with representatives from your user community to make sure your application not only meets the needs of the processes for which it will be used but also meets your users' experience needs. Again, this is particularly critical in a mobile environment, where there is a limited

screen size and a limited amount of viewable data. Important elements of the functional specification are data schemas, screen mockups, and application flows. The functional specification documentation can be used to confirm with project sponsors and users that your application will meet their expectations.

### **Data Schemas**

One approach that helps in mobile development is creating schemas. Schemas capture the data that will be available to the mobile application you are developing. The first step is to create a complete schema that includes all of the data elements of the back-end system with which your mobile application will integrate. Work with management and users to remove from the schema any superfluous information; that is, anything not required by mobile users or not suitable for the mobile environment. Include only those data fields that management and users indicate are absolutely necessary. Cluttered screen arrangements make mobile applications hard to use, as do designs in which multiple screens are required to enter and display the information for a single task. Again, you must be careful to resist the impulse to simply extend a desktop application to a mobile device. Rather, think in terms of providing the minimum functionality necessary to help your users accomplish the task at hand with what is available in your mobile environment.

### **Screen Mockups**

Another necessary tool in mobile application development is screen mockups. These mockups are based on the data schemas, and they allow both developers and users to see what the application will look like before it is developed, in addition to allowing you to experiment with changes in the information used and displayed by the application. Because screens sizes on mobile devices are quite small, mockups are essential to test different data layouts and find the one best suited to a particular task. Mockups can also be used as a way to “lock down” functionality, by getting sign-off from sponsors that what is contained in the mockups is all of the information required to perform the new mobile process. This helps reduce scope creep and rework, too, because it is clear to users and management before development begins exactly what your end product will look like.

## Application Flows

A final tool is an application flows map. This map shows the order in which screens are accessed within the mobile application. Application flows should be designed to fit comfortably with the business processes your application is enabling. Mobile application flows that fail to follow the same sequence as the process flows your users are executing on a desktop/laptop system can result in reduced efficiency and a diminished user experience. The small screens of mobile devices make it easy for your users to get lost in the application. Maintaining as close to a linear application flow as possible reduces the chance of navigational errors.

Force flows are a special type of application flows. They guide your users through the process step by step, removing direct user control of application navigation. If a branch flow is required, the force flow will branch based on conditions defined by the developer. Users are not allowed navigation control; hence, they cannot easily get lost in the application. Force flows are most useful when you are mobilizing well-defined processes that are repeatedly performed, especially by less technical workers.

## 14. Appendix B Mobile Application Development Reference

### 14.1 Mobile Application Architectures

When you are designing mobile applications, there are four main approaches to consider: native clients, Java Platform Micro Edition (Java ME) clients, Web-based clients, and middleware-based clients. A native client is usually written in a lower level language such as C or Assembly and targeted to a specific hardware platform. A Java ME client is written in Java and compiled to run against a Java virtual machine (VM) specifically designed for handhelds and mobile clients. A mobile Web client is very similar to a standard Web client, except that you must carefully consider the layout of the page. Finally, a middleware client utilizes a set of tools and runtimes to abstract the application and data acquisition tasks away from any one device.<sup>3</sup> It is important to understand the benefits and drawbacks of each approach to determine which methodology to choose.

#### Native Client

A native client is written in a lower level language such as C or Assembly and compiled into machine language for a specific group of mobile processors and hardware configurations, and then runs as native code on those mobile devices. The primary benefit to this approach is the ability to utilize to the fullest extent all features of any given hardware. You may need to take this approach, for instance, if your mobile device has specialized hardware that can only be accessed using a C API. Another benefit of developing a native client is that you can tune performance based on needs. For instance, the device may need to perform complex calculations on large datasets. By writing native code, not only is there no abstraction layer converting the client code into native code, but the loops, memory management, and data accesses can be finely tuned to achieve the highest level of performance. All of this power and control, however, comes at a price. Your development must typically target devices that employ similar hardware. This means that you may need to maintain several versions of the source code for the entire set of supported clients. In addition, lower level languages are not usually as productive, so the amount of code you must write to perform a given operation is usually higher. For instance, when using C, you must do all memory allocations manually. This adds not only complexity and

---

<sup>3</sup> Note: Some middleware clients or libraries can be preloaded and are available to applications via API calls during runtime, while others involve compiling the libraries with the application and then loading the resulting executable and required libraries onto the device.

additional lines of code but also, possibly, bugs. In short, you can write a native client when full hardware utilization and the highest performance are the key project factors. When supporting a large number of different mobile clients or when development time is a key consideration, writing a native client is not necessarily the best choice--unless it is your only option.

### **Java ME**

A Java ME client is written in Java and runs using the Java VM on your handheld device. The key benefits to this approach are faster development time and having the same code base target a larger number of devices. Because the application is written in Java, the amount of code required is typically less than it is in a lower level language like C. The reason is that the Java VM automatically handles many of the tedious operations, such as memory management. This, in turn, means that fewer lines of code are required to perform the same business operation. It also means that there are, typically, fewer defects. In addition, because the Java VM itself is available on hundreds of different handhelds, your client application will work on many devices without having to maintain multiple versions. However, because of implementation differences, you will still need to test your application on each platform.

The main drawbacks to the Java ME approach are performance and flexibility. If your application must fit within a small memory footprint or perform frequent operations that are highly CPU (central processing unit) intensive, Java may not be suitable. In addition, if there is a handheld feature that your application must take advantage of but that the Java VM does not expose, you might need native code instead. It may also be possible that one or more of the application's target devices have not had a Java VM ported to them. Ultimately, if your application has a standard interface and does not require special hardware access, Java ME is a good choice. If the client requires the highest level of performance or must utilize specialized hardware, however, Java ME might not be the best choice.

### **Web-Based Client**

A Web-based client runs within a Web browser on the device. The mobile client accesses the page using a browser installed on the device, and the Web server delivers the page using the same facilities as a desktop Web client. The main benefit of this approach is simplified client maintenance, because most handheld devices today have a Web client. This makes it easy to deploy and maintain the software as features and versions change. It also means that as defects are fixed and new features are added, the only deployment is on the Web server.

However, you need to write the application so that it is compatible with your targeted set of browsers. Handheld browser clients are typically not as rich as their desktop counterparts. For instance, DHTML (dynamic hypertext markup language) is limited and sometimes unsupported, frame borders typically cannot be controlled, and multiple windows are not supported, all of which prevent pop-ups and new targets. During the design and specification phase of development, you should take the time up-front to ensure that the targeted browser supports your planned features.

Also, with the proliferation of Web 2.0 features into the mobile handheld world, you can now more easily create richer and more dynamic clients. The main drawbacks to a Web-based client are performance, features, and connection model. With a Web client, the handheld device must have a connection to the Web server to effectively use your mobile application. This means that if your client application needs to perform offline work that can be batched and sent to central servers only a few times a day, a Web client is not a good choice. In addition, if your application requires an extremely rich and dynamic user interface or access to hardware or special I/O (input/output), you may need to consider an alternate implementation. To summarize, if your client software has a simple user interface that can maintain a connection to perform useful work, a Web client is a good choice. Otherwise, an alternate approach will most likely work better.

### **Mobile Middleware**

Another way to deliver functionality to your mobile environment is using mobile middleware. This approach has the primary benefit of rapid development and limited custom and device-specific code to maintain. In the middleware approach, you create an application by utilizing a set of design tools for a third-party, proprietary runtime operating system on the mobile device, in addition to typically utilizing related server-side components. The middleware handles things such as displaying screens and dialogs on the device, caching client-side data, managing the device's connection state, and managing offline data.

The middleware approach works best when your application needs to access and change data from a central server. Typically, you can manage and deploy applications quite quickly. The drawback is the potentially narrow focus of capabilities and the inability to utilize specialized hardware functions on the devices. Middleware platforms also generate code that is not usually suitable for high-performance situations. Ultimately, the middleware approach is very similar to the Java ME approach, only much more specialized, proprietary to the vendor, and generally built around database-driven applications.

In summary, multiple development choices are now available to create mobile applications. Each of the choices has its benefits and drawbacks that will help determine which development approach best meets an application's needs. If your application requires high performance or hardware-specialized code, a native application might be the right choice. If the client needs to run on a wide array of devices and perform offline data manipulation, the middleware approach might be best. If it is a simple application where operation can depend on a connected state, a Web-based client is a good choice. Finally, if your application requires a specialized user interface or specialized client-side code but does not require the full power of a native application, a Java ME application is a good choice.

The following table summarizes the pros and cons of the different approaches.

**Table 3: Pros and Cons of Different Development Approaches**

Architecture	Pros	Cons
Web Browser	Fast to develop No client code to maintain Web 2.0 methods available Works with large range of mobile devices	Less responsive and capable than native applications Less capable than native applications
Java ME	Same code base can support multiple applications Increasing sophistication	Some limits on capability (e.g., no multitasking) Requires testing/adaptation for target platforms
Mobile Middleware	High level of capability with reduced development effort	Additional licensing fees Potentially large learning curve and integration effort
Native Client	Greatest application sophistication and control of local environment Multitasking capabilities on many platforms	Highest level of development effort Different code bases for different devices

When developing mobile applications, there are some additional architectural considerations, as described in the next two sections.

### **Store-and-Forward (Transactional) Mobile Solutions**

One consideration is when your application employs a store-and-forward capability. This approach is also called a transactional solution, because the application queues transactions and delivers them when a connection is next available. Store-and-forward allows users to access the application anywhere, regardless of cellular signal strength or availability at the work location. These systems also tend to offer better perceived performance, because they are not bound by the speed of the network. The mobile application stores data locally on the device and keeps it fresh via periodic background synchronization. The application can synchronize on a schedule, or the server can push new information when it is available.

A transactional solution is less sensitive to fades and drops in the radio signal, because it is neither real time nor session oriented. Most mobile middleware solutions provide smooth recovery if a connection is lost and require no user intervention. Even if the connection is lost in mid transaction, no data is lost, because the application protocols know the transaction state. The application completes the transaction when the connection is next available. The best applications in this category can resume from the point where the connection was lost.

Developing store-and-forward applications using native development tools is challenging, as is maintaining them with the continual introduction of new devices. A commercial point solution or middleware platform is the easiest, fastest, and most reliable way to implement a transactional mobile application.

### **Rich vs. Thin Client Application Deployment Models**

Finally, with a client/server model, you must also consider how much functionality to implement in the device versus how much to implement in the server.

#### **Rich Client**

- Definition: Most significant work done on device
- Device: Smartphone or PDA with sufficient processor/memory resources, screen size, battery, and higher input/navigation features
- Developer environments: Windows Mobile, Java ME, Symbian, Palm, and so on
- Target usages: With applications like customer relationship management (CRM), users send and receive fewer characters, which saves time. Better for data intensive applications

### Thin Client

- Definition: Most significant work done on server
- Device: Internet phone with sufficient network bandwidth
- Developer environments: Wireless Application Protocol (WAP), HTML (hypertext markup language) browsers, Web services, Flash Lite, and so on
- Target usages: Host session can be held open, making it easy for users to resume simple tasks or review data. Easier to deploy information

## 14.2 Mobile Application Development Technologies

In the past few years, the Web has become more of an application platform and less a static display of content. The group of technologies and concepts that enables this is popularly known as Web 2.0. For developers and application builders, this means using Asynchronous JavaScript And XML (Ajax) and DHTML to create applications that can run within a Web browser. Along with Ajax, many other key technologies have emerged, among them: Web services using Simple Object Access Protocol (SOAP) and JavaScript Object Notation (JSON). Both of these technologies help to greatly simplify the interaction of clients and servers, thus facilitating the creation of Web 2.0 applications. In addition, advanced middleware components and tools are available that allow you to create advanced applications quickly and easily. This section will discuss the latest application methodologies and their implications to mobile development.

Two key technologies enable rich application development within the Web browser: Ajax and DHTML, though these are often grouped together in the single term "Ajax." The only thing required to implement an Ajax-enabled Web page is a single API through the XMLHttpRequest object or an ActiveX control for Internet Explorer combined with some DHTML. A properly architected application can significantly reduce the amount of bandwidth consumed by a mobile application while increasing its usability. This is because the server can return a response to the client that does not contain the entire page wrapper. These are typically SOAP or JSON responses that are used to update the contents of widgets in the user interface.

For instance, you may have an application with a selectable list of employees. When a particular employee is chosen, instead of having the browser fetch an entirely new page based on the selected user, the page could generate an Ajax

request, call a corporate Web service, and use the response to dynamically update widgets on the page using DHTML. This not only reduces the amount of data transmitted but also increases the application's responsiveness. Opera, Internet Explorer, Safari Mobile, and Nokia S60 mobile browsers all support Ajax and DHTML. As such, you can now target nearly every device with your mobile applications.

For transporting data to and from the server, mobile applications typically use SOAP or JSON formatted text. Because JSON requires the least amount of work on the client side and is very lightweight, it has become quite popular. A JSON response is a JavaScript object formatted as plain text and then parsed and instantiated using the 'eval' method. For example, you can turn the string '{ "EmpLname" : "Smith", "EmpFname" : "Mike" }' into a JavaScript object by doing `var obj = eval(empJson);`. At this point, it is easy to update an edit or text field with `txtLnameField.Text = obj.EmpLname;`. So, with just three lines of code on the client side, you can convert a once static page into a dynamic application. Clients and servers also use SOAP transporting data, but this is a more complex mechanism. It is a well-formed XML (eXtensible Markup Language) schema and is much more verbose than JSON, but it also allows clients and servers to transport data that may be difficult with JSON. The previous example would expand to the following as a SOAP response:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <ns1:getUserInfo xmlns:ns1=http://domain/userInfo SOAP-
ENC:root="1">
      <UserInfo>
        < EmpLname xsi:type="xsd:string">Smith</ EmpLname >
        < EmpFname xsi:type="xsd:string">Mike</ EmpFname >
      </ns1:getUserInfo>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

As you can see, the SOAP response is significantly larger than the equivalent JSON response. Applications may sometimes require it; for instance, when objects have circular references or when non-JavaScript clients require the same server-side API. Another advantage of using SOAP in your mobile applications is that many more tools are available to create and consume APIs.

The following example illustrates how you might implement a forms-based application using Web 1.0 techniques. In this example, users must first select the state they live in. When they press 'Next,' a new screen appears with a drop-down menu listing a few of the major cities in that state.

**Figure 4: Web 1.0 Example**

**Screenshot 1**



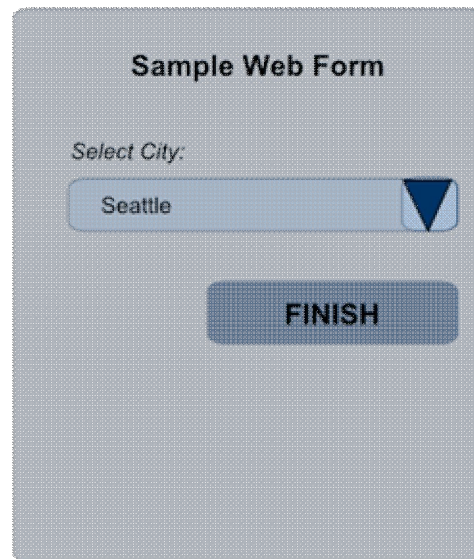
Sample Web Form

Select State:

WA-Washington

NEXT ->

**Screenshot 2**



Sample Web Form

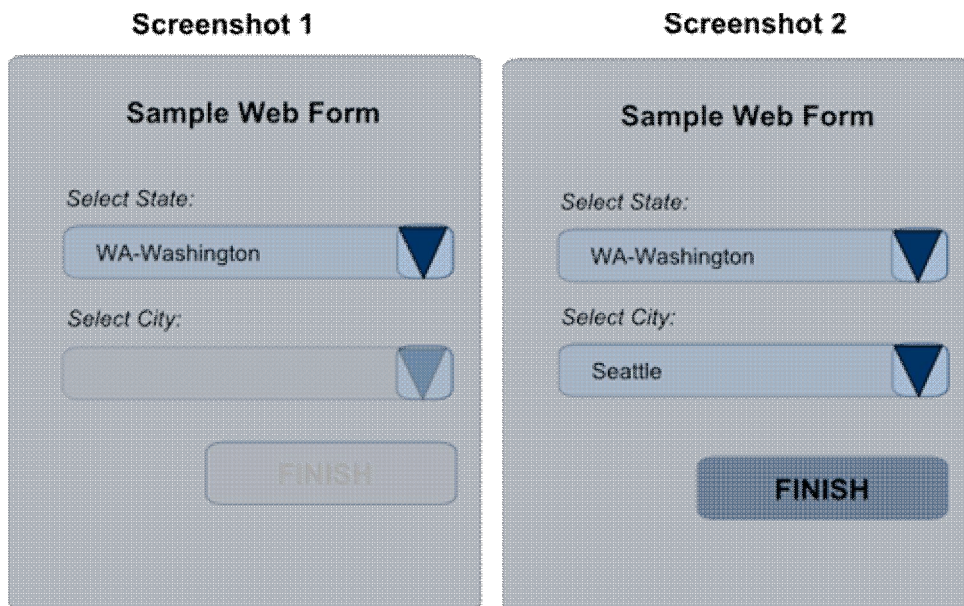
Select City:

Seattle

FINISH

Using Web 2.0 techniques, you would implement the same application as follows:

**Figure 5: Web 2.0 Version of the Same Application**



In the case of the first application, when the user clicks the 'Next' button, the browser will issue a POST request to the Web server, which returns the entire page. Not only does this increase the amount of data transferred, but the user experience is greatly diminished. In the Web 2.0 example, the user would select a state from the drop-down control. This would initiate an Ajax request, which the Web server would respond to with a small JSON response that might look like this: `{ cities : ["Seattle", "Bellevue", "Tacoma", "Everett", "Spokane"] }`. Not only is the amount of data transferred back to the application reduced significantly, but the usability of the application greatly increases. The Web 2.0 application does not force the browser to redraw the entire page, users do not have to click the back button if they wish to select a new state, and the state change of the 'Finish' button readily indicates that users are finished filling out the form. These are some of the key benefits your applications will gain when they are designed for Web 2.0.

Aside from advancements in browser-based applications, there are also numerous middleware platforms that you can leverage to quickly and easily create data-driven applications. Many of the mobile middleware applications narrow the gap between Java ME/C++ applications and Web 2.0 applications by making it easier to create rich graphical user interface (GUI) applications that can utilize lower level features such as subscriber identity module (SIM) and radio access. For instance, M-Business Anywhere from Sybase allows you to create

applications that have the ability to cache and keep synchronized a remote copy of data on the mobile device without writing any code. In addition, you can control the GUI layout of a single application on multiple devices, so you do not have to create unique templates or generate custom GUI code for every device you wish to support. You can also use common standards like Web services and XML within your applications to ensure compatibility across application bases. The drawback to this simplicity is that you must deploy the application on each mobile device that wishes to use it. With a Web-based application, deployment is not necessary, because that occurs as the user accesses the Web page.

### 14.3 Mobile Application Development Tools

The following table identifies the major mobile platforms along with the development environments and languages that are most popular for each platform. Although the list is not exhaustive, it is a good starting point in determining where to begin with mobile development for a particular platform.

In addition to tools for specific platforms, the table also lists tools that support multiple platforms. Note that you can use Sun's Java Wireless Toolkit with many popular integrated development environments (IDEs), including Eclipse and NetBeans. Many of the listed development toolsets are not simply IDEs; they are also sets of independent tools that can assist in your mobile development.

**Table 4: Mobile Platform Development Matrix**

OS	Language	Development Toolset	URL
Symbian	C/C++	Nokia Carbide Development Tools for Symbian OS C++	<a href="http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide/">http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide/</a>
	Java	Nokia Carbide Development Tools for Java - Carbide.j	<a href="http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide/">http://www.forum.nokia.com/main/resources/tools_and_sdks/carbide/</a>
	C/C++	Wirelexsoft VistaMax	<a href="http://wirelexsoft.com">http://wirelexsoft.com</a>
	C/C++	ARM RealView Development Suite	<a href="http://www.arm.com/products/DevTools/RealViewSoftwareDevelopment.html">http://www.arm.com/products/DevTools/RealViewSoftwareDevelopment.html</a>
	C/C++	SymbDev	<a href="http://pf128.krakow.sdi.tpnet.pl/symbdev/">http://pf128.krakow.sdi.tpnet.pl/symbdev/</a>
Windows Mobile	C/C++	Visual Studio 2005 + Windows Mobile SDK	<a href="http://www.microsoft.com/windowsmobile/developers/">http://www.microsoft.com/windowsmobile/developers/</a>
	.NET	Visual Studio 2005 + Windows Mobile SDK	<a href="http://www.microsoft.com/windowsmobile/developers/">http://www.microsoft.com/windowsmobile/developers/</a>

OS	Language	Development Toolset	URL
Palm	C/C++	Garnet OS Development Suite	<a href="http://www.access-company.com/developers/downloads/palmostools.html">http://www.access-company.com/developers/downloads/palmostools.html</a>
	C/C++	PRC Tools	<a href="http://prc-tools.sourceforge.net/">http://prc-tools.sourceforge.net/</a>
BlackBerry	Java	BlackBerry Java Development Environment	<a href="http://na.blackberry.com/eng/developers/downloads/jde.jsp">http://na.blackberry.com/eng/developers/downloads/jde.jsp</a>
All	C/C++	Eclipse CDT	<a href="http://www.eclipse.org/cdt/">http://www.eclipse.org/cdt/</a>
	Java	Sun Java Wireless Toolkit 2.5.1	<a href="http://java.sun.com/javame/index.jsp">http://java.sun.com/javame/index.jsp</a>

## 14.4 Capability Summary of Mobile Platforms

When creating mobile applications, you need to carefully consider the capabilities of the mobile devices on which your application will execute. The capabilities of devices vary tremendously, and it is likely that your application will support only specific devices.

The following table lists the types of capabilities available on mobile devices, along with general comments.

**Table 5: Different Capabilities of Mobile Devices**

Capability	Comment
Wireless Networking	Some devices support only EDGE, others EDGE and Broadband Connect. Not all areas have Broadband Connect.
Operating System and Tools	Smartphone platforms provide tools for mobile application development in high-level languages such as C and Java.
Java Capability	Most devices today support Java Platform Micro Edition.
Browsers	Browser capabilities vary by device. For example, only some support AJAX.
Location Capabilities	Some devices now have location capability.
Keyboard	Varies by device. Some have full QWERTY keyboards, some have predictive text capability for keypads, some only have touchscreens.
Local Storage	Varies by device. Some have optional memory expansion slots providing for up to 4 GBytes with today's capability, and much higher amounts planned for the future.
Memory Volatility	Most devices today have non-volatile memory.
Computational Power	Varies by device. Test computationally-intensive applications on the slowest target device.

## 14.5 Throughput and Latency

When you develop mobile applications, it is important to consider the capabilities of your wireless network. Today's 3G networks have broadband capability with user achievable throughput rates often over 1 megabit per second (Mbps). However, the actual throughput a user will experience depends on a variety of factors, including network loading and signal quality. It also depends on whether you are connecting via EDGE (available nationwide) or BroadbandConnect (available in many metropolitan areas.) In addition, you will need to know the capabilities of your target devices.

BroadbandConnect is AT&T's 3G network operating on GSM™, the worldwide standard for wide-area wireless communication. BroadbandConnect was the first widely available service in the world to use High-Speed Downlink Packet Access (HSDPA), and it is the only 3G technology that supports simultaneous voice and data.

The following table summarizes downlink throughput speeds. Uplink speeds vary by device, but they are generally slower than downlink speeds.

**Table 6: Wireless Network Throughput Speeds**

Technology	Typical Downlink Throughput Rates	Burst Speeds
EDGE	75 to 135 kbps	200 kbps
BroadbandConnect	400 to 700 kbps	1 Mbps

Note that these values will continue to increase as AT&T makes improvements to its network. See the white papers at 3G Americas (<http://www.3gamericas.com>) that describe the evolution of capabilities with Third Generation Partnership Project (3GPP) technologies.

Latency is another consideration. Wide-area wireless networks have higher latency than wireline networks.

AT&T recommends that you:

1. Understand whether the device will support just EDGE or also BroadbandConnect

2. Develop for conservative data throughput
3. Minimize the number of back-and-forth messages
4. Manage carefully the amount of data your application transmits and receives
5. Test your application in multiple locations to verify expected application transaction speed
6. Test in both Second Generation (2G)--for example, EDGE--and 3G--for example, BroadbandConnect--coverage areas

## 14.6 Battery and Power Management

Mobile devices have constrained battery life. As such, your mobile applications should accommodate limited battery life and be able to recover from any sudden loss of power.

The items that consume the most power on mobile devices include the following:

1. Backlight
2. Voice activity
3. Data transmission
4. Data reception
5. Intensive computational activity

An application cannot necessarily control how much a user interacts with the device, which will keep the backlight on, but good user interface design can reduce the amount of time a user needs to spend executing different application operations.

With regard to data communications, sending data from the device consumes more power than receiving data. An efficient design that minimizes the amount of data communicated will also minimize costs on usage-based data plans.

Another common problem is loss of power. Though device battery life is improving, there is always the possibility that a user or an application will not synchronize with a server in time. Application methods should allow graceful recovery, in addition to periodic data synchronization to minimize data loss. Some devices have non-volatile memory, which makes data loss less likely.

One of the difficulties in application development is determining how to make the tradeoff between CPU usage and memory consumption. Unfortunately, you must typically choose one or the other. With mobile development, the situation is even more sensitive, because after the radio and the display, the CPU consumes much of the battery power. For this reason, you must consider how your application algorithms will affect battery usage.

For example, if you are tasked with writing an image manipulation program that requires a complex matrix manipulation on each bit of the image, you could implement this as a CPU-intensive algorithm or as a memory-intensive algorithm. The program could perform the required matrix algebra on the entire image every time a user selects a particular operation, with the result stored in memory. Or, the operation could be performed for only the viewable portion of the screen, with updates done as required. The first method requires a significant amount of memory, but it might end up using fewer CPU cycles—especially if the same sections of the screen are frequently redrawn. The second method requires constant extra CPU cycles as the field of view on the image changes. Ultimately, the best choice will be the algorithm that consumes the least CPU cycles while not utilizing so much memory that the application cannot run.

Another example of where a CPU algorithm might be used instead of a memory algorithm is compression. If your application manipulates very large documents or objects, you may have no choice but to use compression in memory to give users access to the entire document. If, however, your application can fit objects into memory without compression or cache them efficiently to persistent storage, overall CPU usage will most likely decrease and battery life will increase.

As opposed to desktop computing environments, you must carefully consider how your application algorithms will affect power consumption. The general rule is to cache as much data as possible without using so much memory that your application will no longer run. Another consideration: Just because an algorithm uses more memory, does not necessarily mean that it will use fewer CPU cycles. In the image-viewing example above, the method where the updated image is cached may actually require more CPU cycles if a majority of the image is never viewed.

Finally, you should be prepared to test more than one algorithm and to structure your application code in such a manner that it can be easily swapped for a more efficient algorithm in the future.

## 14.7 Input, Text, Screens, Usability

Handheld devices present dramatically different user interfaces than desktop PCs. The best handheld applications fully accommodate the more constrained display and limited data input capability of smaller devices. Your application design should consider the following specific items:

- **Application scope.** Consider carefully just how much data mobile users need. Often, this may be a subset of what they would use in an office environment.
- **Small displays.** Smartphones today typically have screens of 240x320 or 320x320 pixels--only a small percentage of a desktop or laptop screen. Carefully decide what information you want to present. Be aware that different target devices may have different screen sizes. Because of the small screen size, only one application typically owns the screen, and it is not possible to have multiple windows open.
- **Limited data input.** Even with QWERTY keyboards, users type at slower speeds on mobile devices than they do with a regular keyboard. Minimize how much data users have to enter by using predictive methods such as auto-text completion and multi-choice selections and by presenting previous entries.
- **One-handed operation.** Handhelds are often used in a one-handed fashion, especially when users are standing or walking. Consider designing your application so that screens can be navigated and all desired information viewed without using the keyboard or stylus.
- **Exploit navigation options.** Many handhelds have soft keys, four-way navigation buttons, and trackballs. These options allow users to easily reach all desired information and features, especially one-handed.
- **Background processing and communication.** Design your application so that transactions (such as communications with a server) occur in the background, do not hold up the user interface, and allow continuous use of other operations.
- **Progress indicators.** Where appropriate, provide users with an indication of communications status; for example, how many records have been received out of how many are being sent.
- **Ability to cancel.** Users sometimes need to suddenly turn off their devices; on an airplane, for instance. Where appropriate, provide users

with the ability to abort operations, especially those that may take a lot of time.

## 14.8 Memory Management

Although the amount of memory available on mobile devices has expanded over the last few years, it is still highly constrained compared to the desktop environment. It is, therefore, important to consider these constraints when designing and developing applications to ensure your application performs well. This section discusses some of the programming approaches that address memory constraints.

Depending on the environment, you can apply different techniques to ensure that your application will run in a constrained memory environment. In the Java ME and Web 2.0 worlds, no explicit memory management is available; however, the method you use to code certain algorithms can greatly impact memory usage. For instance, if your application reuses variables within a loop instead of declaring new instance variables, it may reduce memory consumption by a significant degree.

One simple programming technique a Java developer can use to show a list of names is to store all of the names in an array and access different indices in that array as the user scrolls the cursor up and down. The primary advantages to this approach are performance and simplicity. It performs well, because all of the memory accesses come from a linked list and only require a single dereference to find the structure in memory. And it is simple, because the Java ME runtime library creator has already written the inner workings of the linked list, such as `ArrayList`. Therefore, you need only use the simple API methods of 'add' and 'get' to do most of the work. But what if there are 100,000 items to show in the linked list? At some point--typically not a very large number--the mobile device will run out of memory to allocate to the application. In the case of Java ME, an `OutOfMemoryException` will be thrown. Several possible solutions are available to overcome this limitation.

One option is to trade space for time. This means using more CPU cycles to overcome the environment's memory limitations. The most common way you can achieve this is through persistent storage caching. In the case where the `ArrayList` needs to contain 100,000 items but the device does not have the memory to contain all of these items, you can choose to use three `ArrayLists` of 1,000 items each. One list could be the current list, one could be the pre-list, and

one could be the post-list. The current list is the one being used to draw the user interface. The post-list is the pre-loaded list of 1,000 items that come after the current list. When the user scrolls the display forward, the application checks to see if it is at the end of the current list. If it is, the application makes the current list the new pre-list and makes the post-list the new current list. It then spawns a thread to load the new post-list. You would reverse these steps as the user scrolls in the opposite direction. With this approach, you need to write more code and potentially spend more CPU cycles loading objects. But it also allows your application to scroll through an indefinite number of objects and only requires that, at most, 3,000 items be loaded into memory versus 100,000.

Another time-space tradeoff you can use is memory compression. Depending on the format and type of data, your application may benefit greatly from compressing data in memory and only decompressing what it needs for display or current manipulation. For instance, your application may need to display and scroll large map surfaces for the user to add pinpoints, pan, or zoom. In this case, you could keep the large version of the map in a highly compressed format, perhaps a meta or vector format such as SVG (scalable vector graphics) or even a raster format such as PNG (portable network graphics) or JPEG (joint photographic experts group). As the user moves the map around the mobile device's display, your application could calculate the bitmap to display on the screen while keeping a majority of the image compressed. In a desktop environment with nearly unlimited memory, it might be unwise to make an application decision like this, but in a mobile environment it may be the only way to make your application operational.

In unmanaged environments such as C++, not only are memory algorithms important but so are proper memory allocation and freeing. As opposed to Java or JavaScript, where a runtime garbage collector specifically frees and reuses memory that is no longer being referenced, an application in an unmanaged environment must explicitly free all memory. This is perhaps the single largest class of errors that C and C++ programmers make. The primary reason is that it can quickly become unclear, because of function calls, pointer copying, and vague APIs, who has responsibility to free chunks of memory. This means that an application or library that might run fine on a desktop environment could end up crashing after you port and run it on a mobile device.

The primary methods available to find and fix memory leaks are using memory debugging facilities and writing test cases for your public class methods and functions. Most software development kits (SDKs) ship with libraries that contain memory debugging APIs. For instance, on the Symbian operating system, if you

build your application in debug mode, an exception will be thrown from the debug heap manager informing you that it found memory leaks. You can also generally uncover memory leaks and programming errors by writing test cases that force the most extreme use of your public functions.

By paying close attention to memory management, you will be able to create powerful and robust mobile applications.

## 14.9 Network

How your application interacts with and accommodates the wireless network is of crucial importance. A number of items make wide-area wireless different than the other types of networks with which you may be working.

First, user accounts are provisioned with particular networking configurations that control the types of Internet Protocol (IP) addresses assigned (static versus dynamic, public versus private), what external networks can be accessed, and security options. These configuration settings are called Access Point Names (APNs). Wireline networking has no equivalent, though corporate network configurations and Internet service provider (ISP) policies/options have some corresponding characteristics. APN security options are discussed in the “Security” section of Appendix B.

Second, devices establish data sessions through what is called a Packet Data Protocol (PDP) context. Data sessions are independent of voice capability, and they require a separate procedure typically controlled by the mobile application via networking APIs. PDP context activation results in the mobile device receiving an IP address and being able to send and receive IP packets. One important aspect of PDP contexts is that the network times them out if there is no activity. The timeout value is approximately four hours if the network can reach the device and about one hour if the network cannot communicate with the device.<sup>4</sup> Connection management is discussed in greater detail the “Connection Management” section of Appendix B.

Third, how your company’s servers communicate with the AT&T network (to communicate with mobile devices) must be considered. AT&T offers a number of networking options. The default is communication via the Internet. However, AT&T also offers a network VPN for IPsec-based communication across the Internet and Frame Relay connections that bypass the Internet.

---

<sup>4</sup> AT&T can customize these values for custom APNs.

The following table summarizes the key differences between wireless and wireline networking.

**Table 7: Wireless vs. Wireline Networking**

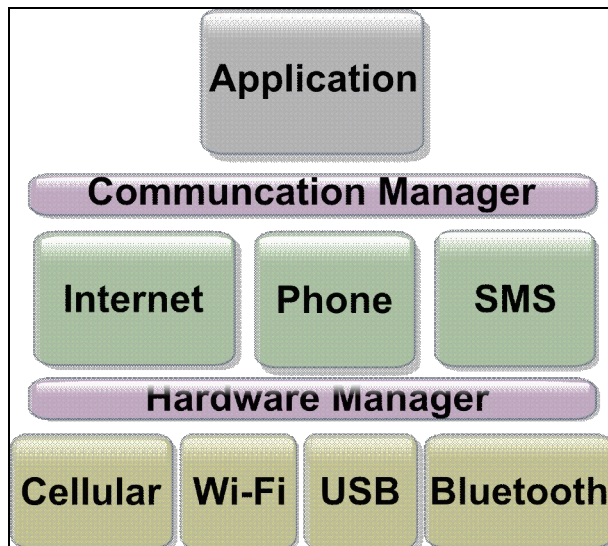
	<b>Wireless</b>	<b>Wireline</b>
<b>Access Point Names</b>	With EDGE and 3G, specifies items such as type of IP address, mobile termination, security options, fixed-end connectivity	ISP policies and account options provide some, but not identical characteristics
<b>Data Sessions</b>	Created through Packet Data Protocol (PDP) context  These time out with inactivity	Either always available or created via Point-to-Point session with ISP  IP addresses usually assigned by Dynamic Host Control Protocol (DHCP)
<b>Fixed-End Connectivity</b>	AT&T offers Internet, VPNs over Internet and Frame Relay	ISPs typically offer Internet

The Network section of AT&T's devCentral site has extensive details on all of these topics.

## 14.10 Connection Management

New mobile devices typically have a number of connection choices, including cellular data service, Bluetooth, Wi-Fi (Wireless Fidelity), and USB (Universal Serial Bus). Although this greatly enhances the usability and mobility of the devices, it complicates the process of determining which connections are available and best to use. Fortunately, nearly all mobile operating systems and platforms supply APIs that abstract the network hardware and software layers, in addition to simplifying the process of connection management, as shown in the following figure.

**Figure 6: Handheld Device Communication Architecture**



The following table lists the key platforms and their connection management APIs.

**Table 8: Some Common APIs for Connection Management**

Platform	API	Reference URL
Windows Mobile	Connection Manager C API	<a href="http://msdn2.microsoft.com/en-us/library/bb416435.aspx">http://msdn2.microsoft.com/en-us/library/bb416435.aspx</a>
Palm	Garnet OS NetLib	<a href="http://www.access-company.com/developers/documents/palmos/palmos.html">http://www.access-company.com/developers/documents/palmos/palmos.html</a>
Symbian	Connection Manager C++ API	<a href="http://www.symbian.com/developer/techlib/v7_0sdocs/doc_source/reference/cpp/ConnectionManager/RConnectionClass.html">http://www.symbian.com/developer/techlib/v7_0sdocs/doc_source/reference/cpp/ConnectionManager/RConnectionClass.html</a>
Java ME	Generic Connection Framework javax.microedition.io	<a href="http://java.sun.com/javame/reference/apis/jsr218/javax/microedition/io/package-summary.html">http://java.sun.com/javame/reference/apis/jsr218/javax/microedition/io/package-summary.html</a>

The first thing an application needs to do to send or receive data is establish a connection. Some of the platforms will do this automatically, while others require the application to create the connection. For example, in Windows Mobile, you call `ConnMgrEstablishConnection` to set up a connection. You should also check the connection status—to make sure the connection is currently available—before any network operations using the connection are performed. Thus, the connection can be reestablished, if necessary, without an application failure.

Your applications should also release their connections when they are no longer needed, so that the connection manager can properly manage the network layer connection. Sometimes, however, it may be desirable to not release the connection; if, for instance, it will be needed in the near future. This will keep the device from having to reestablish a network layer connection, which could slow down application performance.

One common pitfall is that developers create their applications assuming they are always connected—like Ethernet. AT&T is making investments to increase coverage and improve session persistence, but you must assume that an application will sometimes go out of coverage.

## 14.11 Security

Securing your mobile devices and applications requires a number of important decisions, ranging from protection against eavesdropping and user authentication to protection against device loss and threats to devices (such as viruses). A good starting point is to understand what security options AT&T and your network provide, what third-party security tools and application are available, and what options exist for your actual application.

### Network Security

The AT&T network provides a number of security features and options. The first is authentication against the credentials in the SIM. This prevents unauthorized users from accessing a user's account for voice or data service. With GSM/EDGE, the network authenticates the user device (that is, the phone or data card). With Universal Mobile Telecommunications System (UMTS)/HSDPA, on the other hand, there is two-way authentication in which the network authenticates the device and the device authenticates the network.

The AT&T network also encrypts data communications using 64-bit encryption for EDGE and 128-bit encryption for UMTS/HSDPA. Encryption extends from the

mobile device to the core network. Within the core network, however, user data is unencrypted (unless encrypted by the customer) while data flow is across a private network.

In addition, AT&T has firewalls within its network that implement various security procedures. These include blocking unsolicited IP packets from the Internet so that they do not reach mobile devices and blocking direct device-to-device communication. With custom APNs, enterprise customers can customize firewall rules to accommodate specific needs.

### **Third-Party Security Options**

Various third-party security options exist for handheld devices. First, VPNs can secure communications and provide user authentication. The two most common types of VPNs--IPsec and SSL--can operate on handheld devices. However, not all IPsec-based VPNs provide clients for mobile devices. SLL VPNs are easier to deploy, because they can work with most handheld browsers. Mobile VPNs are also optimized for mobile/wireless communications and offer additional benefits such as session persistence. Windows Mobile platforms tend to be most widely supported by VPN vendors.

Beyond VPNs, companies also offer management systems that include security options like being able to delete data on a lost or stolen device and disabling the device entirely.

Finally, most mobile middleware platforms provide robust security features.

### **Application Security**

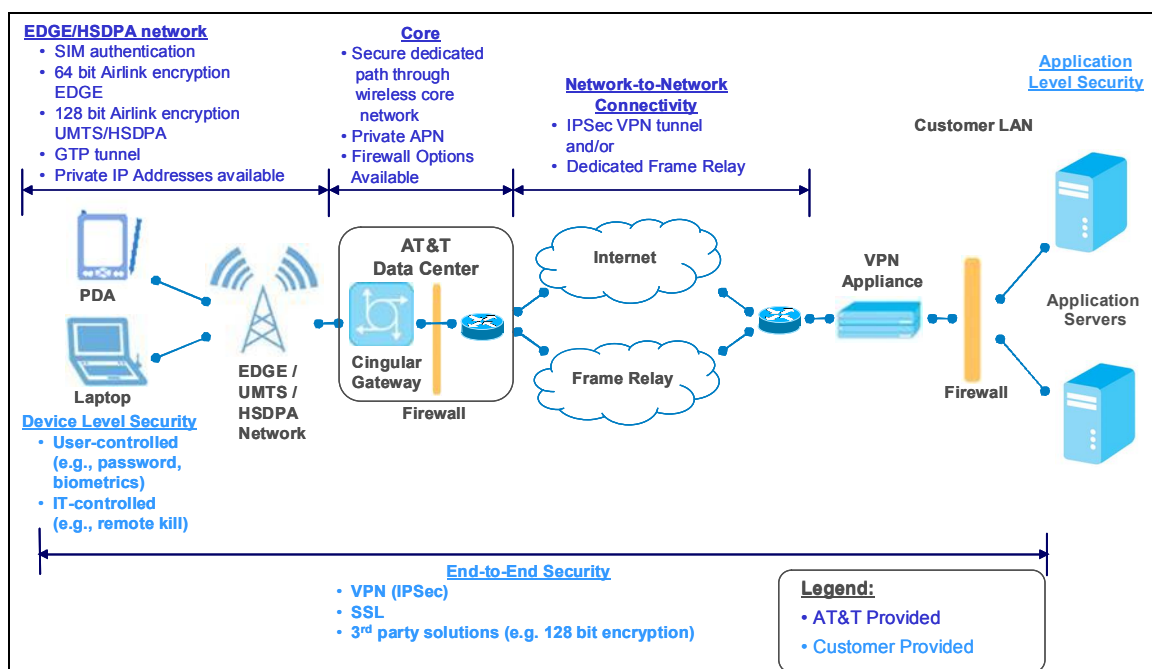
The security options listed above may be sufficient, or you may decide that your applications may need to implement additional security functions. For instance, you may want to encrypt sensitive data on mobile devices. Most handheld platforms provide cryptographic libraries that can be integrated into your mobile applications.

If you are not using a VPN, you may want to protect your data communications. Options include encrypting the data prior to sending it or leveraging the SSL layer in the Web browser and sending the data through an HTTPS (Hypertext Transfer Protocol over Secure Sockets Layer) session. Because HTTPS is a resource-intensive protocol, you should use it appropriately.

If you wish to employ biometric security, refer to devCentral at:  
<http://developer.att.com/biometrics>.

The following diagram shows the different locations where either the network or the customer implements security.

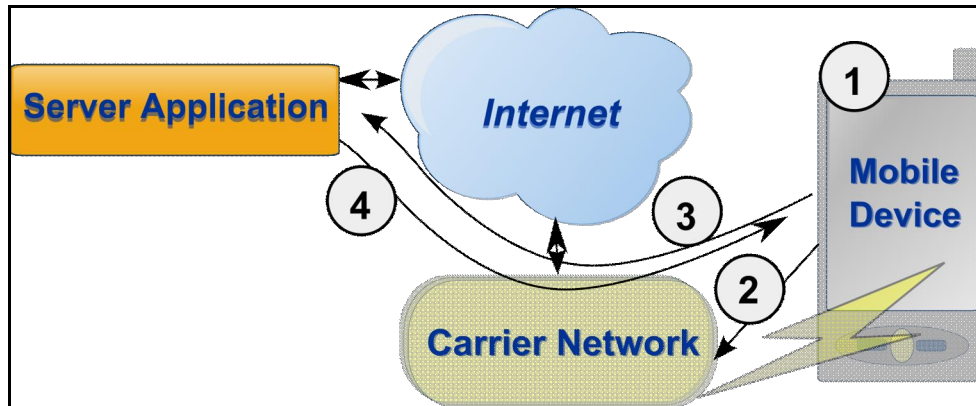
**Figure 7: Security on an End-to-End Basis**



## 14.12 Push vs. Pull

Most mobile applications broadly fit one of two service models: push or pull. In the push model, new events, messages, or data are sent to the mobile device as they occur without the device first requesting them. In the pull model, the mobile device makes a request for data or events. The following diagram illustrates a typical pull architecture.

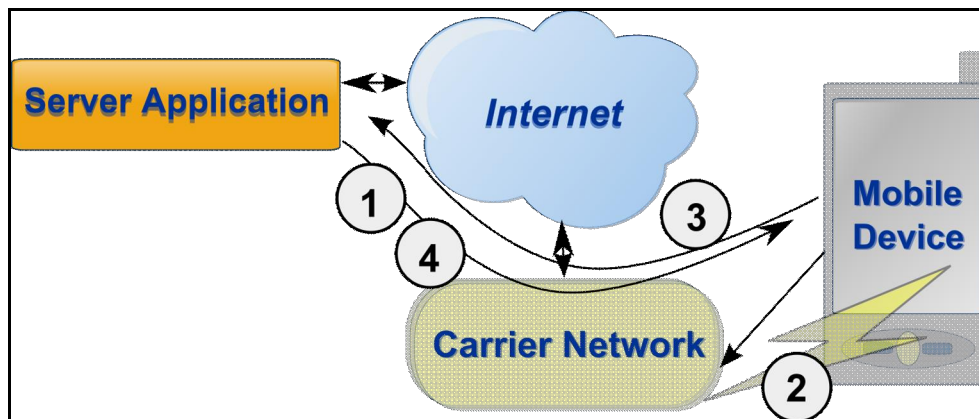
**Figure 8: Pull Architecture**



1. User initiates transactions (via user interface, timer, application event)
2. Device establishes transport connection (EDGE, Wi-Fi, and so on)
3. Device application establishes connection to server application, sends request
4. Server responds with data

The following diagram illustrates a typical push architecture.

**Figure 9: Push Architecture**



1. An event occurs on the server that the mobile device should be notified about. If the server does not have a connection to the device already, it

can attempt to initiate one by sending an SMS to the mobile device or waiting for the device to reestablish a connection

2. Device establishes transport connection (GPRS, Wi-Fi, and so on), if one does not already exist
3. Device establishes a connection to the server application and sends a request for the corresponding event
4. Server sends response data

One common method that applications use to implement a push architecture is through the use of SMS. When the server needs to update the device, it can submit an SMS message to the device using the Short Message Peer-to-Peer Protocol (SMPP). When the device receives the SMS message, the application can then obtain the update via an IP data connection. An application using Java Mobile Information Device Profile (MIDP) can use the Java ME PushRegistry API to facilitate this approach.

Other than using SMS for notifications, a common way applications can maintain connection states through which servers can push information is a constant PDP context. This requires the device application to establish an initial connection with the server, so that the server knows the IP address of the mobile device, and requires the mobile application to monitor its connection and IP address. If the connection fails, or the IP address changes at any time, the mobile application must reestablish the connection and notify the server of its new IP address.

For instance, using the Java MIDP API, you could establish a TCP connection by calling `Connection.open` (e.g., `socket://mycompany.com:18865`). A service listening on this port could then track mobile devices and make notifications when appropriate. When the mobile application receives a notification, it could then call `Connection.open` (<https://mycompany.com/getData.jsp>) to obtain the updated data.

Because the default APN from AT&T does not allow mobile-terminated IP packets, you will need to use an APN that allows such communication. You will need to carefully design your application protocols, so that they do not consume excessive data in maintaining the connection. Finally, this architecture increases the amount of power consumed by the device.

The following table outlines the benefits of the push and pull architectures.

<b>Pull Benefits</b>
Client and server are both simpler to engineer
Typically consumes less bandwidth
Allows the user to determine when bandwidth is used
<b>Push Benefits</b>
The user is notified in near real time when an event occurs
Can reduce bandwidth consumption if events are infrequent <sup>5</sup>

Which approach is best depends on your application. Push applications are significantly more complex to engineer, but they can provide significant benefits when the value of real-time information--such as some e-mail or financial information--is extreme.

### 14.13 Telephony, SMS, SIM Interfaces

One area unique to mobile application development is the use of the telephony, SMS, and SIM APIs that are available on these platforms. These APIs allow a user to make calls, send SMS messages, or look up entries stored on the phone's SIM card. By using these functions, you greatly simplify application development that requires phone-specific functionality. You can also use these APIs to increase the functional capabilities of your mobile application.

For instance, you may want to create an application that notifies other mobile users when an event occurs. To perform this operation, your application could use the SIM APIs (assuming the SIM was used to store telephone numbers) to determine where to send the message and then to actually send the message. Telephony APIs allow your application to make and receive phone calls.

The following table lists the key platforms and links to their telephony, SMS, and SIM APIs.

---

<sup>5</sup> For instance, if you received two e-mails per day, a push solution would consume very little bandwidth compared to a pull solution, where the mail application polls on a regular interval throughout the day.

**Table 9: SMS and SIM APIs for Handheld Platforms**

Platform	API	Reference URL
<b>Windows Mobile</b>	Telephony	<a href="http://msdn2.microsoft.com/en-us/library/aa918634.aspx">http://msdn2.microsoft.com/en-us/library/aa918634.aspx</a>
	SMS	<a href="http://msdn2.microsoft.com/en-us/library/aa922463.aspx">http://msdn2.microsoft.com/en-us/library/aa922463.aspx</a>
	SIM	<a href="http://msdn2.microsoft.com/en-us/library/aa919162.aspx">http://msdn2.microsoft.com/en-us/library/aa919162.aspx</a>
<b>Palm OS</b>	Telephony	<a href="http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/TelephonyBasic.html#1044083">http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/TelephonyBasic.html#1044083</a>
	SMS	<a href="http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/TelephonySMS.html#1056053">http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/TelephonySMS.html#1056053</a>
	SIM	<a href="http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/TelephonyPhonebook.html#1070579">http://www.access-company.com/developers/documents/docs/palmos/PalmOSReference/TelephonyPhonebook.html#1070579</a>
<b>Symbian</b>	Telephony	<a href="http://www.symbian.com/developer/techlib/v8.1adocs/doc_source/guide/Telephony-subsystem-guide/N1012E/index.html">http://www.symbian.com/developer/techlib/v8.1adocs/doc_source/guide/Telephony-subsystem-guide/N1012E/index.html</a>
	SMS	<a href="http://www.symbian.com/developer/techlib/v9.2docs/doc_source/reference/reference-cpp/MSG_SMS8.1/index.html#MSG_SMS8%2e1%2etoc">http://www.symbian.com/developer/techlib/v9.2docs/doc_source/reference/reference-cpp/MSG_SMS8.1/index.html#MSG_SMS8%2e1%2etoc</a>
	SIM	<a href="http://www.symbian.com/developer/techlib/v9.2docs/doc_source/reference/reference-cpp/PHBKSUNC/index.html#PHBKSUNC%2etoc">http://www.symbian.com/developer/techlib/v9.2docs/doc_source/reference/reference-cpp/PHBKSUNC/index.html#PHBKSUNC%2etoc</a>
<b>Java</b>	Telephony	<a href="http://java.sun.com/products/jtapi/">http://java.sun.com/products/jtapi/</a>
	SMS	<a href="http://java.sun.com/products/wma/index.jsp">http://java.sun.com/products/wma/index.jsp</a>

## 14.14 Communication Cost Management

When designing the communications aspects of your application, you should consider the pricing plans your users are on. Some plans support unlimited data usage (with some restrictions), but others are usage based. Lower amounts of data usage could translate to lower monthly fees. (More efficient data communication also extends battery life.)

During development it may be a good idea to measure the amount of data your application consumes. Unfortunately, there are no readily available third-party utilities that run on mobile devices available for this task. However, ample

network monitoring tools are available, and you can install these tools on the server side of the connection. Wireshark, formerly known as Ethereal, is an open-source tool that can be used to monitor all of the traffic on a network. Microsoft's NetMon is a helpful tool specific to Windows. And AT&T provides daily data consumption for user accounts on its bills and user-account Web pages through the previous day.

The following section discusses how developers can manage the volume of communicated data.

## 14.15 Managing Amount of Data Communicated

Even though mobile networks have become much faster, applications should still be designed to fully optimize how, and even if, large amounts of data are sent. You can use many standard techniques to minimize the amount of data your application sends. The first, and perhaps simplest, method is to compress the data. The next method is differencing and caching. The final method is data conversion or reduction. Data conversion and caching are typically more complex than compression, but these options may yield larger benefits. The following paragraphs cover these methods in greater detail.

Applications frequently use compression to maximize limited RAM (random access memory) and persistent storage. This can be equally important when bandwidth is limited. If the data your application manipulates is sparse or repeating, it will benefit significantly from compression. Examples of this are large plain text files such as XML documents, some raw sound file types like WAV (Waveform audio format), or flat graphics data such as bitmap. The following table lists some of the compression APIs readily available for common platforms.

**Table 10: Compression APIs for Handheld Platforms**

Platform	API	Reference URL
<b>Windows Mobile</b>	System.IO. Compression	<a href="http://www.microsoft.com/downloads/details.aspx?FamilyID=1343d537-a62f-4a6e-9727-7791bf4cc2bd&amp;displaylang=en">http://www.microsoft.com/downloads/details.aspx?FamilyID=1343d537-a62f-4a6e-9727-7791bf4cc2bd&amp;displaylang=en</a>
	zlib	<a href="http://www.zlib.net/">http://www.zlib.net/</a>
<b>Palm OS</b>	Lz77 API	<a href="http://www.access-company.com/developers/documents/palmos/palmos.html">http://www.access-company.com/developers/documents/palmos/palmos.html</a>
<b>Symbian</b>	Syslibs EZLIB	<a href="http://www.symbian.com/developer/techlib/v8.1adocs/doc_source/reference/reference-cpp/N102C2/CEZFileToGZipClass.html">http://www.symbian.com/developer/techlib/v8.1adocs/doc_source/reference/reference-cpp/N102C2/CEZFileToGZipClass.html</a>

Platform	API	Reference URL
Java	java.util.zip	<a href="http://java.sun.com/j2se/1.5.0/docs/api/java/util/zip/package-summary.html">http://java.sun.com/j2se/1.5.0/docs/api/java/util/zip/package-summary.html</a>

The next method is differencing and caching. Depending on the data, this method can be simple or very complex. One example is Concurrent Versioning System (CVS) and diff. In the early years of the Internet, when links were significantly slower and storage much more expensive, the Unix diff application was developed to eliminate sending redundant data. This application compares two files and creates a third file that is the difference between the two. CVS then uses this difference to update the current version of a file. At this point, another developer could use CVS to update a local copy with the much smaller diff file, thus producing the new file. The idea behind this technique is that not all of the data is sent; instead, only what has changed is passed. For instance, if your mobile application works on a large XML document, it might be much more efficient to send only the items the user has updated on the device. You can also use this method in conjunction with compression to possibly increase efficiency further. The main drawback to this approach is that your application must track what data has changed, which sometimes can be difficult.

Another common technique is to reduce or distill the document format of the data used. For example, if you must send an image to the mobile device, it might not make sense to send the full resolution image if it cannot be manipulated on the device. In this case, the size of the sent image could be reduced to match the size of the device display. You can also use this method for documents that only have a limited subset of the features available on the mobile device. PDF (Portable Document Format) converters that only transfer the text of PDF documents and spreadsheet applications that do not embed charts and graphics are examples.

## 14.16 Recovery and Diagnostics

Mobile applications can fail in various ways. This section discusses the failure modes and provides suggestions for recovery and diagnostic mechanisms.

Failure modes include:

- **Signal loss.** Users can sometimes find themselves suddenly out of their coverage area. This can occur at any point in mid-transaction. Well-

designed applications either detect connectivity loss or time out gracefully and then resume transmission once the signal is regained.

- **Poor quality signal.** With a poor quality signal (low signal strength or high interference environment), data communication may still be possible, but it may occur with very low throughput rates and much higher packet delays. Application protocols may time out. TCP protocols may also operate in a suboptimal fashion.
- **Connection timeout.** The AT&T network maintains data connection states (called PDP context) for up to four hours when in coverage and up to one hour when out of coverage. After a PDP context timeout, the user must reacquire a data connection to send or receive data. Usually, this PDP context will require a new IP address.
- **Loss of power.** The mobile device's battery may suddenly run out in mid-transaction or with data files open. Or, the user may suddenly turn off or reset the device.

It can be challenging to debug networking applications in general, and wireless ones in particular, because of additional potential failure modes. The following types of recovery and diagnostic mechanisms may be helpful:

- **Display transaction state.** Have the application provide status information about its networking operations.
- **Log communications sessions.** On either the device side or the server side, keeping a log of communications sessions can help determine the sources of failures. A diagnostic mode might capture entire session contents.
- **Provide reset options.** Sometimes it can be beneficial for the application to restart the data connection or even the entire radio connection. This can be automatic after a certain number of communication attempts or an option that lets users easily restart their sessions.
- **Provide an alternate network path on failure to reach host.** The APN determines the transport for data communication between the wireless device and the host application. A wireless device can have access to multiple APNs<sup>6</sup>, which can be selected by the application in the event one of the available transports fails. Multiple transports, such as Frame

---

<sup>6</sup> As part of account provisioning.

Relay and network VPNs, in conjunction with a custom APN can provide redundancy in the configuration.