devCentral

Palm Wireless Application Development

White Paper

Document Number	**12899**
Revision	**1.0**
Revision Date	**07/17/03**

AT&T Wireless

Legal Disclaimer

Revision History

Date	Revision	Description
07/17/03	1.0	Initial release of document.

Table of Contents

Table of Contents

Table of Contents

1. Introduction

This paper explains how developers can design applications that operate on the Palm OS platform and communicate using the AT&T Wireless (AWS) General Packet Radio Service (GPRS) network.

Since Palm, Inc. (Palm) and other parties provide thorough documentation on developing Palm OS applications, this paper emphasizes aspects unique to wireless networking.

This paper includes an overview of wireless options for Palm OS devices, development tools, programming considerations, and sample wireless applications.

1.1 Audience

This paper has been developed for independent developers, enterprise developers, AWS Alliances, and system integrators engaged in application development. The paper assumes a working knowledge of GPRS and reasonable knowledge of Palm OS programming.

1.2 Contact Information

If you have comments or questions regarding the information within this document, please e-mail them to developer.program@attws.com . Please reference the name of this document in your e-mail.

Authors:

Eric Lyons, Objectware (http://www.objectwareinc.com)
Peter Rysavy, Rysavy Research (http://www.rysavy.com)

1.3 Resources

This section summarizes resources, including those available from Palm, useful books, developer tools, papers, and specifications.

1.3.1 Palm, Inc.

This subsection lists the resources available to developers from Palm.

Palm OS Software Developer Kit (SDK)
http://www.palmos.com/dev/tools/sdk

This SDK provides the libraries and development tools to develop wired and wireless Palm OS applications for both PRC Tools and Metrowerks CodeWarrior.

Palm OS Developer Documentation
http://www.palmos.com/dev/support/docs/

The following developer documents and sections at this Palm site highlight wireless application development:

- **Palm OS Programmer's Companion, Volume II**
  http://www.palmos.com/dev/support/docs/palmos/Companion2TOC.html

  The "Network Communication" section introduces the Net Library, Internet Library and SSL Library. The "Internet and Messaging Applications" section introduces Web Clipping and sending e-mail via the default e-mail application. The "Telephony Manager" section introduces the use of the Telephony API to manage telephony capabilities, short message service (SMS) and the subscriber identity module (SIM) module.

- **Palm OS Programmer's API Reference**
  http://www.palmos.com/dev/support/docs/palmos/ReferenceTOC.html

  Part III outlines each function in the Net Library API and Telephony API. Part IV outlines each function in the Internet Library API.

- **Palm OS Programmers API Reference—Telephony SMS**
  http://www.palmos.com/dev/support/docs/palmos/TelephonySMS.html

- **Palm OS Programmers API Reference—Telephony Calls**
  http://www.palmos.com/dev/support/docs/palmos/TelephonyCalls.html

- **Palm OS Emulator (POSE)**
  http://www.palmos.com/dev/tools/emulator

  This application emulates Palm hardware on Windows, Macintosh and Linux desktops to facilitate debugging with Palm OS development tools.

The Palm OS Emulator emulates devices with version 4.1, or earlier versions of the Palm OS that use Motorola 68000 based processors.

- **Palm OS Simulator**
  http://www.palmos.com/dev/tools/simulator

  This application simulates Palm hardware on Windows desktops to facilitate debugging with Palm OS development tools. The Palm OS Simulator simulates devices with version 5.0 or newer versions of the Palm OS that use Advanced RISC Machine (ARM) processors.

- **Web Clipping Application Development**
  http://www.palmos.com/dev/tech/webclipping/

  This includes an overview, technical documentation and development tools for creating Web Clipping applications.

- **Developer Forums**
  http://www.palmos.com/dev/support/forums/

  These developer forums offer a place to ask development questions and to review message archives for Palm OS development. You can access forums via e-mail or by configuring a newsgroup reader.

The following forums are geared specifically towards networking and wireless application development. You can use the escribe.com links to browse forums archives without subscribing to the forum:

- **Communications Forum**
  Archive: http://www.escribe.com/computing/communicationforum
  Forum: news://news.palmos.com/comm-dev-forum

- **Web Clipping Forum**
  Archive: http://www.escribe.com/computing/pqaforum
  Forum: news://news.palmos.com/pqa-dev-forum

- **Handspring Developers Resource Page**
  http://www.handspring.com/developers/develop.jhtml

  This site includes links to Handspring newsgroups and white papers pertaining to Handspring's GPRS/GSM API and libraries, data communication with Handspring devices and using the SMS library.

1.3.2 Books

The following lists useful books for Palm platform software development.

Palm OS Programming: The Developer's Guide
Authors: Neil Rhodes, Julie McKeehan
Publisher: O'Reilly
ISBN: 1-56592-856-3 (Second Edition, October 2001)
http://www.oreilly.com/catalog/palmprog2

Palm OS Network Programming
Author: Greg Winton
Publisher: O'Reilly
ISBN: 0-596-00005-7 (First Edition, September 2001)
http://www.oreilly.com/catalog/palmosnetpro

Palm OS Web Application Developer's Guide
Authors: Ben Combee, R. Eric Lyons
Publisher: Syngress
ISBN: 1-928994-32-6 (First Edition, 2001)
http://www.syngress.com/catalog/sg_main.cfm?pid=1392

Piloting Palm
Author: Andrea Butler, David Pogue
Publisher: John Wiley & Sons
ISBN: 0471089656 (First Edition, February 2002)
http://www.pilotingpalm.com/

1.3.3 Developer Tools

The following lists the primary developer tools available for the Palm platform.

CodeWarrior for the Palm OS
Publisher: Metrowerks, a Motorola company
http://www.metrowerks.com/MW/Develop/Desktop/PalmOS/Professional

PRC-Tools: GCC (GNU Compiler Collection) for the Palm OS
Publisher: SourceForce.net
http://prc-tools.sourceforge.net

AppForge MobileVB
Publisher: AppForge, Inc.
http://appforge.com/prod/af_prof/index.html

Satellite Forms MobileApp Designer
Publisher: Pumatech, Inc.
http://www.pumatech.com/sf_mad_main.html

1.3.4 Browser Software

The following lists the browsers for the Palm platform.

AvantGo
Publisher: AvantGo
http://www.avantgo.com

Blazer 2.0
Publisher: Handspring
http://www.handspring.com/blazer

NetFront v3.0
Publisher: Access Systems America, Inc.
http://www.access-us-inc.com/products/cewb_nf.asp

Palm WAP Browser
Publisher: Palm, Inc. and AU-System
http://www.palm.com/support/handbooks/tungstent/wapbrowser_hb.pdf

Palm Web Browser Pro 1.0
Publisher: Palm, Inc.
http://www.palm.com/software/webbrowserpro

PalmSource Web Browser 2.0
Publisher: Palm, Inc.
http://www.palmos.com/dev/tech/palmos5/webbrowser.html

1.3.5 Specifications and Papers

The following lists the technical specifications and papers related to the topic of this white paper.

"General Packet Radio Service (GPRS); GPRS Ciphering Algorithm Requirements"
Publisher: 3rd Generation Partnership Project (3GPP)
http://www.3gpp.org/ftp/Specs/html-info/0161.htm

"Technical Realization of the Short Message Service (SMS)"
Publisher: 3rd Generation Partnership Project (3GPP)
http://www.3gpp.org/ftp/Specs/html-info/0340.htm

"General Packet Radio Service (GPRS), Service Description"
Publisher: 3rd Generation Partnership Project (3GPP)
http://www.3gpp.org/ftp/Specs/html-info/0360.htm

"Use of Data Terminal Equipment—Data Circuit Terminating Equipment (DTE-DCE) Interface for Short Message Services (SMS) and Cell Broadcast Services (CBS)"
Publisher: 3rd Generation Partnership Project (3GPP)
http://www.3gpp.org/ftp/Specs/html-info/0705.htm

"AT Command Set for GSM Mobile Equipment (ME)"
Publisher: 3rd Generation Partnership Project (3GPP)
http://www.3gpp.org/ftp/Specs/html-info/0707.htm

"Specification of the Subscriber Identity Module—Mobile Equipment (SIM-ME) Interface"
Publisher: 3rd Generation Partnership Project (3GPP)
http://www.3gpp.org/ftp/Specs/html-info/1111.htm

White Paper: Handheld Security for the Mobile Enterprise, Palm Inc., September 2002.
http://www.palm.com/pdfs/handheld_security.pdf

1.4 Glossary

Table 1 defines terms and acronyms used in this document.

Table 1 Terms and Acronyms

Term or Acronym	Definition
ARM	Advanced RISC Machine
CHTML	Compact HTML
CSLIP	Compressed SLIP
CSV	Comma Separated Value
DNS	Domain Name System
DHCP	Dynamic Host Configuration Protocol
DTMF	Dual Tone Multi Frequency
EMC	Emergency Management Center

Term or Acronym	Definition
FTP	File Transfer Protocol
GCC	GNU Compiler Collection
GEA	GPRS Encryption Algorithm
GGSN	GPRS Gateway Support Node
GNU	GNU’s Not Unix
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HDML	Handheld Device Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
INetLib	Internet Library
IP	Internal Protocol
NetLib	Net Library
PKI	Public Key Infrastructure
PPP	Point To Point Protocol
PRC	Palm Resource
RISC	Reduced Instruction Set Computer
SD	Secure Digital
SGSN	Serving GPRS Support Node
SIM	Subscriber Identity Module
SLIP	Serial Line Internet Protocol
SMTP	Simple Mail Transfer Protocol
SMS	Short Messaging Service
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VPN	Virtual Private Network
WAP	Wireless Application Protocol
WML	Wireless Markup Language
XML	Extensible Markup Language
xHTML	Extensible Hypertext Markup Language

2. Device Categories

2.1 GSM Device Categories

Numerous Palm OS handheld solutions are available for wireless applications using GSM/GPRS networks. This section discusses the pertinent aspects of the different device configurations.

2.1.1 Integrated GSM/GPRS

Devices in this category have a GSM radio built-in for phone calls and GPRS data access, as shown in Table 2.

Table 2 Palm Devices with Integrated GSM/GPRS

Device	Product Information URL
Handspring Treo 180	http://www.handspring.com/products/treo/index.jhtml
Handspring Treo 270	http://www.handspring.com/products/treo270/index.jhtml
Palm Tungsten W	http://www.palm.com/products/handhelds/tungsten-w

Note: Not all of these devices are currently available for the AWS GSM/GPRS network.

Integrated devices offer the convenience and simplicity of GRPS data access and phone calls in one device. In addition, connection and network settings are typically pre-configured.

However, some feature sacrifices are made to incorporate a GSM radio into the device. For example, the Handspring Treo line offers no card slot for expansion cards such as memory cards and the screen is smaller than a typical Palm handheld screen. The Tungsten W offers a Secure Digital (SD) slot and a standard size screen. However, it uses an earpiece and does not include a speaker or microphone. Whereas the Tungsten W uses OS 4.1, newer Palm devices are using OS 5 with Advanced RISC Machine (ARM) processors. Nevertheless, these are excellent solutions for customers wishing to use just one device for data and voice.

2.1.2 Bluetooth Connected Phone

Bluetooth is a short-range, low power wireless networking technology for computing devices in a Personal Area Network (PAN). The signal range for Bluetooth radios in Palm handhelds and mobile phones is nominally ten feet, though you may achieve greater distances. The Bluetooth connection can forward network requests from the Palm to the mobile phone. This approach allows most of the same networking applications that operate on an integrated device to also function on the combination of a Palm device and Bluetooth-equipped phone. In this configuration, the Bluetooth phone appears to the Palm device as a modem.

More information about Bluetooth is available from the Bluetooth website at http://www.bluetooth.com/

The Palm devices listed in Table 3 have built-in Bluetooth.

Table 3 Palm Devices with Integrated Bluetooth

Device	Product Information URL
Palm Tungsten T	http://www.palm.com/products/handhelds/tungsten-t
Sony NZ90 CLIÉ	http://www.sonystyle.com/clie
Sony TG50 CLIÉ	http://www.sonystyle.com/clie

For Palm devices that do not have built-in Bluetooth networking, you can add Bluetooth using the Palm Bluetooth card. Table 4 shows examples of such devices.

Table 4 Palm Devices that Accept a Palm Bluetooth Card

Device	Product Information URL
Palm m130	http://www.palm.com/products/palmm130
Palm m515	http://www.palm.com/products/palmm515
Palm i705	http://www.palm.com/products/palmi705

Product information for the Palm Bluetooth Card is available at http://www.palm.com/products/accessories/expansioncards/bluetooth

To use Bluetooth, you need to combine the Palm device with a GSM phone that has integrated Bluetooth support. AWS currently offers the Bluetooth phones listed in Table 5.

Table 5 AT&T Wireless GSM Phones with Bluetooth Capability

Mobile Phones	GSM Bands
Nokia 3650	http://www.nokiausa.com/phones/3650
Nokia 6310i	http://www.nokiausa.com/phones/6310i

Mobile Phones	**GSM Bands**
Siemens S56	http://www.siemens.com (search for S56)
Sony Ericsson P800	http://www.sonyericsson.com/p800
Sony-Ericsson T68i	http://www.sonyericsson.com/t68i

In addition to data connectivity, the Palm device can use Bluetooth to initiate phone calls on the mobile phone.

There are various advantages of using a Bluetooth connection. By assigning the cellular-data transmission responsibilities to a Bluetooth phone, Bluetooth-enabled Palm handhelds can feature greater functionality and longer battery-life. As the capabilities of Palm handhelds and Bluetooth/GSM phones improve, you can selectively upgrade either device. And because the Palm device can initiate phone calls, you do not need to maintain contact information on the mobile phone.

The disadvantage of a Bluetooth approach is that it adds configuration tasks not present with integrated GSM devices. Before the two devices can exchange data via Bluetooth, you must conduct a one-time pairing that involves interaction with both devices. While not difficult, the pairing process can involve multiple steps and is not always intuitive.

Users must also carry both devices. While the phone can stay in a pocket, purse, or briefcase for GPRS data access, managing two devices can sometimes be cumbersome. Finally, you must maintain power levels for two separate devices, both of which require separate chargers.

3. Connections

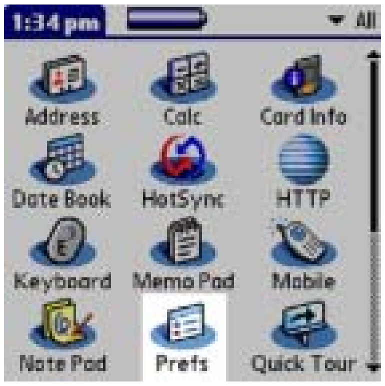
GPRS data sessions consist of IP-based communications. The GPRS modem (whether integrated or in a phone) appears to the Palm device as a modem controlled by AT commands. In establishing a connection, the GPRS network assigns an IP address, similar to any other Internet Service Provider. Once connected, mobile devices can send and receive packets with minimal delay, and appear as IP nodes on an IP network. Since GPRS is a packet-data service, it only consumes the radio resource while sending and receiving, allowing you to maintain extended data sessions even while idle.

A critical aspect of using GPRS is ensuring devices are configured correctly, and for applications to manage the connection, which includes detecting whether a connection is already established, creating the connection to the network if necessary, monitoring the connection, potentially leaving it available for other applications, and tearing down the connection as necessary.

3.1 Device Configuration

The Palm OS has supported connection and network definitions since the introduction of the Net Library in version 2.0 of the Palm OS. The configuration process for GSM networks is similar to the configuration process for wire line modems and dial-up Internet accounts.

Figure 1 Palm Screen Showing the Preferences Application



The Preferences application offers separate dialogs for configuring connections and networks. The Connection dialog defines the physical connections between the Palm OS and other computing devices.

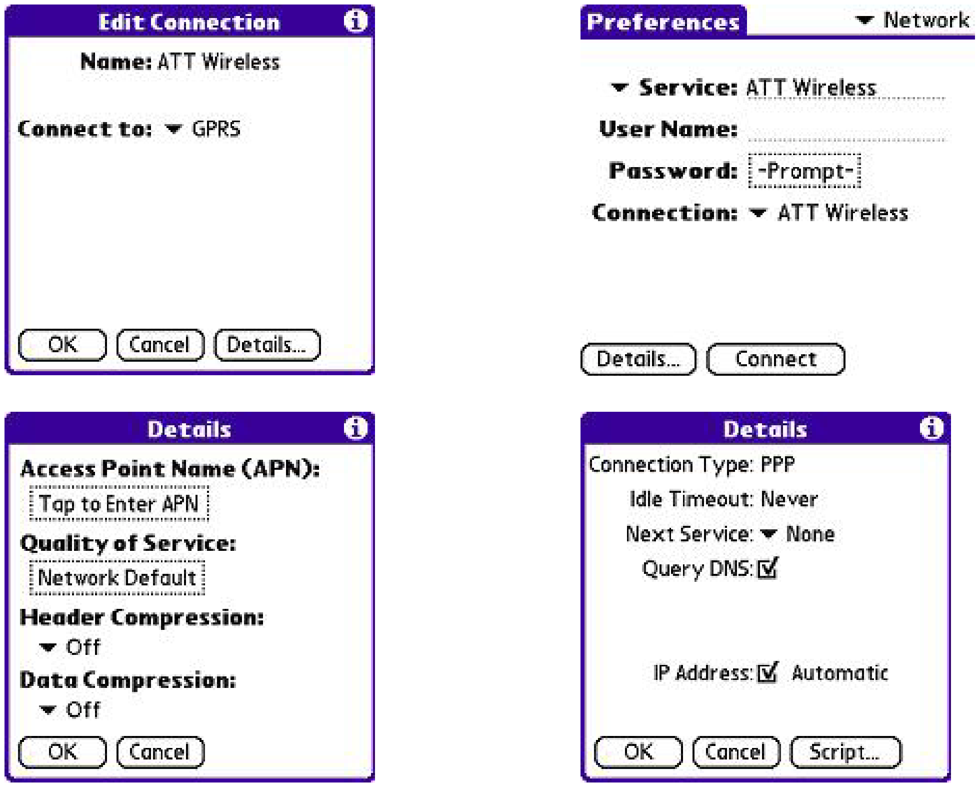
Examples include the synchronization cradle, infrared light, wire line modem and GPRS modem.

The Network dialog defines network communication properties such as the physical connection, data layer (PPP, SLIP, CSLIP), timeouts and login scripts.

3.1.1 Configuration with Built-In GSM Radio

Connection and network configurations are pre-configured for devices with built-in GSM radios. For illustration purposes, the following screenshots outline the default connection and network configurations on the Palm Tungsten W.

Figure 2 Palm Tungsten W Edit Connection and Network Configuration



Edit Connection

Name: ATT Wireless

Connect to: GPRS

Preferences — Network

Service: ATT Wireless

User Name:

Password: -Prompt-

Connection: ATT Wireless

Details

Access Point Name (APN): Tap to Enter APN

Quality of Service: Network Default

Header Compression: Off

Data Compression: Off

Details

Connection Type: PPP

Idle Timeout: Never

Next Service: None

Query DNS: ☑

IP Address: ☑ Automatic

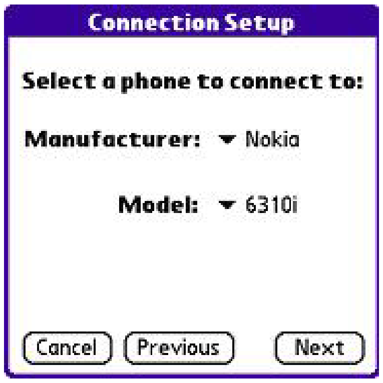
3.1.2 Configuration Using Bluetooth

To use Bluetooth between a Palm device and a GSM phone, you must define a “trusted pair” which bonds the Palm device to the GSM phone. To simplify this process, Palm, Inc. provides a wizard on the “Software Essentials” CD-ROM called “Phone Link” that creates the connection and network configurations. While not a “one-step” process, the most difficult question is usually the make and model of the mobile phone.

The following screenshots show the steps in the Phone Link application to pair a Palm Tungsten T handheld with a Nokia 6310i mobile phone.

Note: Informational dialogs have been removed for brevity.

Figure 3 Steps to Pair a Palm Tungsten T with a Nokia 6310i

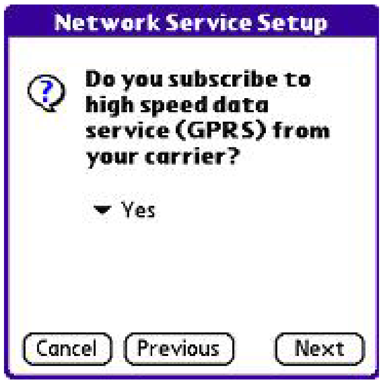


Connection Setup

Select a phone to connect to:

Manufacturer: Nokia

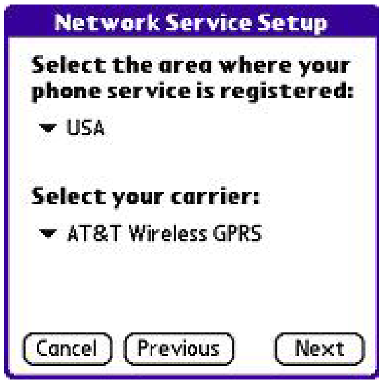
Model: 6310i



Network Service Setup

Do you subscribe to high speed data service (GPRS) from your carrier?

Yes


Network Service Setup

Select the area where your phone service is registered:

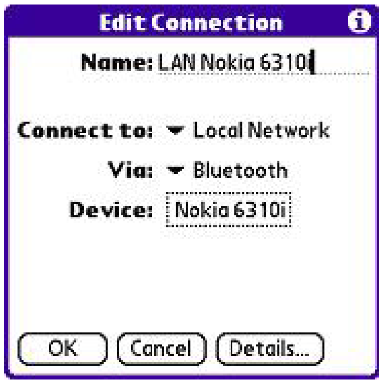
▼ USA

Select your carrier:

▼ AT&T Wireless GPRS

The following screenshots show the default connection and network parameters for a Palm Tungsten T paired with a Nokia 6310i. The first screenshot is the LAN connection configuration between the Tungsten T and the Nokia via Bluetooth. The last two screenshots describe the network configuration using GPRS.

Figure 4 Connection Parameters for an Example Bluetooth Connection

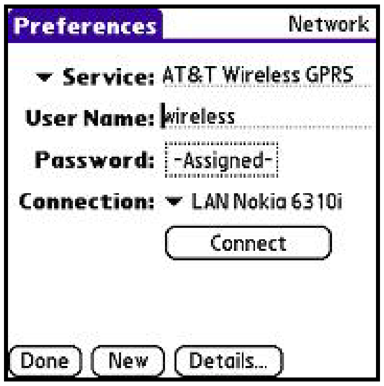

Edit Connection

Name: LAN Nokia 6310i

Connect to: ▼ Local Network

Via: ▼ Bluetooth

Device: Nokia 6310i

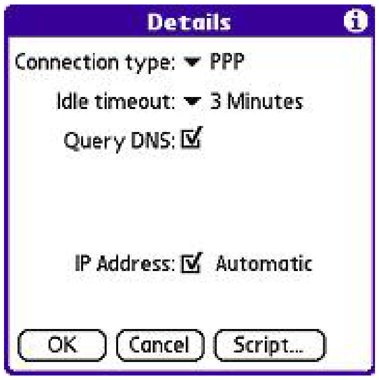

Preferences Network

▼ Service: AT&T Wireless GPRS

User Name: wireless

Password: -Assigned-

Connection: ▼ LAN Nokia 6310i


Details

Connection type: PPP

Idle timeout: 3 Minutes

Query DNS: ☑

IP Address: ☑ Automatic

3.2 Connection Management

Once you have configured your device(s) for GPRS network operation, you will need to manage connections. This section discusses how this is done, whether by applications or by the user. It also discusses usage meters so you can monitor the amount of data communicated across the GPRS connection, an important consideration as many service plans are based on the amount of data communicated.

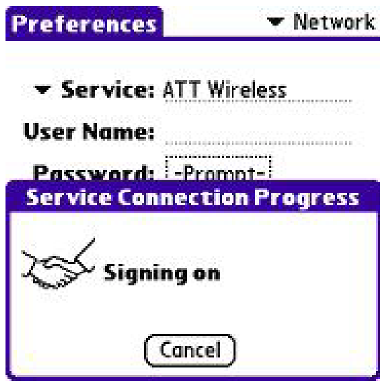
3.2.1 Connecting

When data services are needed by a Palm OS application, the Palm OS uses the properties in one of the network configurations to establish a network connection. The connection type (Point-to-Point Protocol, Serial Line IP), static IP, Dynamic Host Control Protocol IP addressing, and the login script sequence are examples of network configuration properties.

Applications can invoke connections using calls in the Net Library or Internet Library. The following screenshot shows a connection being established.

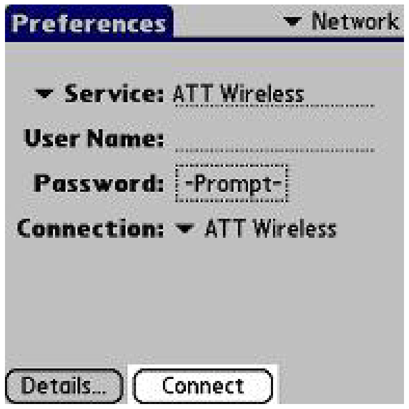
A GRPS network connection is typically established in 10-15 seconds. Progress dialogs are provided by the Palm OS to indicate the progress of the network connection and to report any errors, as shown in the following screenshot.

Figure 5 Connection Progress Dialog, Superimposed Over a Network Dialog



In addition to having applications automatically connect when data service are required, users can manually connect and disconnect from the GPRS network from the network configuration in the Network dialog of the Preferences application, as shown in the following screenshot.

Figure 6 Screen Showing How Users Can Manually Connect



3.2.2 Disconnecting

When data services are no longer needed by the application, the application terminates the GPRS network connection immediately, or preferably uses the “Idle timeout” property in the network configuration to determine when to terminate the connection.

When the application terminates the GPRS network connection immediately, the next GPRS data request must reestablish the network connection. When the application uses the “Idle timeout” property, the GPRS network connection is left active until the timer expires without network activity.

Generally, well-behaved applications will allow the GPRS network connection to remain until the timer expires. Maintaining the connection allows a user to switch between network-capable applications without having to wait the ten to fifteen seconds needed to reestablish the GPRS network connection.

Well-behaved applications should also confirm that the GPRS network connection is still available before trying to re-establish a connection.

The following situations can also terminate a GPRS network connection:

- **Network Inactivity.** The GPRS network terminates a connection after four hours of inactivity or one hour of being out of coverage.
- **Signal Quality.** The GPRS radio may terminate a connection if the signal becomes weak or lost.
- **Power Management.** The Palm OS may terminate the connection if the device runs low on power.
- **User Activity.** The Palm OS terminates the connection when the device is turned off or when the user turns off Wireless Networking.

3.2.3 Usage Meters

The Tungsten W maintains usage meters for GSM phone calls and GPRS data transfer. Totals are set to zero for new devices and reset when a hard reset is performed. Users can monitor usage and reset the meters in the Mobile dialog in the Preferences application.

In a Bluetooth to Bluetooth/GSM mobile phone setup, the mobile phone may provide voice and data usage statistics. In this configuration, the Palm handheld does not maintain these counts.

4. Developing Networking Applications

This section identifies the major choices available for developing networking applications for Palm handhelds. Since GPRS provides an IP networking service, applications use the networking capabilities of the Palm system to communicate with external networks.

Table 6 summarizes the networking technologies available for both Palm devices with integrated GSM capability and Palm devices connecting to GSM/GPRS phones using Bluetooth.

Table 6 Summary of Networking Tools for the Palm Platform

Networking Technology	Integrated GSM (Palm)	Integrated GSM (Handspring)	Integrated Bluetooth	Add-On Bluetooth To Palm Device
Net Library	■	■	■	■
Internet Library	■			■
Network Synchronization	■	■	■	■
Telephony: Voice Calls	■	■ †	■	■
Telephony: SMS	■	■ †	■	■
Telephony: SIM Access	■	■ †	■	■
Web browsers	■	■	■	■
Web Clipping	■			■
Operating System	Palm OS 4.1	Palm OS 3.5.2	Palm OS 5.0	Palm OS 4.1
Devices	Palm Tungsten W	Handspring Treo 180 Handspring Treo 270	Palm Tungsten T Sony NZ90/TG50	Palm m130 Palm m515 Palm i705

■ Indicates application is supported.

† Indicates Handspring provides its own proprietary Voice, SMS, and SIM libraries. The Handspring libraries are similar to the Telephony API libraries introduced in Palm 4.0. You will need to separate the code to support these capabilities on both Palm and Handspring platforms.

In developing communications-oriented applications for the Palm platform, you can consider either native applications that execute locally on the Palm platform, or Web-based applications that take advantage of a browser on the Palm device. The following sections describe these approaches in detail.

4.1 Native Applications

Native applications are compiled binary executables (Palm Resources or PRCs) comparable to EXE files in Windows. You typically develop native applications using C/C++ or the language provided with one of the third-party development tools. There are two principal networking libraries that your application can use: Net Library (NetLib) and Internet Library (INetLib). The following sections examine the applicability and merits of each. With native applications, you can also perform synchronization across the network, control telephony functions, send and receive short messages (SMS), and access the SIM card.

4.1.1 Net Library

The Net Library (NetLib) is a Palm OS system library that provides native applications basic network services using TCP and UDP via a socket API. You can build higher-level Internet-based services (file transfer, e-mail, web browsing, etc.) using NetLib calls.

Palm introduced NetLib with the release of Palm OS 2.0 in 1997 to add networking capabilities to devices with a telephone line modem or a networked cradle.

NetLib has been available in Palm OS devices since the first PalmPilot in 1997. Even if the device does not come with built-in networking hardware, you can use NetLib to control networking peripherals that connect via the HotSync connector or other expansion slots such as Handspring’s Springboard.

The following code demonstrates how you may load the Net library if it is not already loaded.

```
err = (SysLibFind("Net.lib", &m_wLibRef))
if (err != 0)
{
   /* If the library is not loaded, load the library*/
      err = SysLibLoad (sysFileTLibrary, sysFileCNet,
&AppNetRefnum);
   if (err)
   /* display some error for the user */
   throw sysErrLibNotFound;
}
```

Below are some sample API functions in NetLib:

- **NetLibSocketOpen**—Open a new socket.
- **NetLibSocketBind**—Assign a local address to a socket.
- **NetLibSocketListen**—Listen to incoming connections.
- **NetLibSocketAccept**—Accept a connection from a stream-based socket.
- **NetLibSend**—Send data on an active socket via a single buffer.
- **NetLibReceive**—Receive data on an active socket via a single buffer.
- **NetLibSocketClose**—Close the socket.

For a more comprehensive Net Library API listing please refer to:

http://www.palmos.com/dev/support/docs/palmos/NetLibrary.html

The following discussion lists the pros and cons of using Netlib.

Pro: Device, Vendor, and Network Independent

NetLib is available on devices with Palm 2.0 or higher. An application developed with NetLib will run on a wide variety of Palm OS devices.

NetLib functionality is essentially the same for devices from different Palm OS vendors and licensees. Vendors do not create unique implementations of NetLib for their devices.

NetLib is connection and network independent. An application targeted for a GPRS-capable device should also work for devices other wireless networks and wire-line modems for network access.

Pro: Berkeley Sockets Standard

NetLib is based on the Berkeley sockets API specification with which most TCP/IP programmers have intimate knowledge for functions such as socket, connect, read, send and close. Numerous resources are available to learn how to program networking applications using Berkeley sockets.

While Palm uses standard Palm OS naming conventions for NetLib functions and structures, Palm does provide a socket.h header file to map these API to the standard Berkeley names using macros. In essence, you can code and debug networking components of an application as a desktop application. Once complete, you can transfer the code to the Palm application code with few or no modifications.

Pro: Flexibility

The NetLib implementation is not bound to any one protocol or service. You can develop applications NetLib for both the TCP and UDP protocols and services such as FTP, TELNET, SMTP, etc.

Con: Complexity

However, flexibility can translate to complexity. The flexible nature of NetLib requires that you understand the intimate details of Internet protocols and services. You must explicitly code tasks such as domain name resolution and socket management. In desktop environments, these details are often abstracted with higher-level code libraries or controls.

Net Library does not post specific events to the Palm application event loop to signal connection loss or any other NetLibrary specific events. The only way to determine connection "status" for example is via the standard Error value returned by Net Library API functions.

4.1.2 Internet Library (INetLib)

The Internet Library (INetLib) is a Palm OS system library built on NetLib to provide native applications simplified access to high-level Internet protocols such as HTTP and HTTPS for accessing web documents.

Palm introduced INetLib in 1999 with the release of Palm OS 3.2 and the Palm VII, Palm's first wireless device. In addition, Palm introduced the Web Clipping mini-browser technology that was built using INetLib. However, INetLib is not available in OS 5.x.

The following discussion lists the pros and cons of using INetLib.

Pro: Simple HTTP/HTTPS Programming

INetLib simplifies HTTP programming by combining many typical Internet networking steps into one or two INetLib API calls. This reduces your learning curve and the coding effort to write HTTP based applications.

Pro: Content Optimization for Handhelds

Instead of communicating directly to remote servers, INetLib calls communicate via the Palm.Net proxy servers. When an application issues an HTTP request using INetLib, the request is routed to a Palm.Net proxy server. The Palm.Net proxy server contacts the remote host over a wired network to retrieve the response.

Before sending the response back to the handheld, the Palm.Net proxy server optimizes the HTML and images in the response. The document length may be shortened and graphics may be resized. The Palm.Net proxy server may also compress the response for more efficient transmission through the wireless network.

Once the Palm.Net proxy server sends the response back to the handheld, INetLib on the handheld automatically processes the response, including decompressing it if required, and presents the response to the application as if the Palm.Net proxy played no part in the transmission.

Con: INetLib not on All Devices

INetLib is a technology proprietary to Palm, Inc. handhelds. Other wireless device manufacturers such as Handspring have chosen not to include INetLib on their devices and instead use NetLib for networking applications.

For a while, Palm, Inc. provided software called the Mobile Internet Kit that added INetLib to Palm devices and other third-party devices. However, Palm has since removed the Mobile Internet Kit product from its Web site. It appears that Palm favors a complete OS upgrade to version 4.1 to add INetLib.

Con: INetLib not in Palm OS 5.x

Palm, Inc. has not provided INetLib for its latest OS platform, OS 5. It also does not appear that Palm will provide INetLib in future OS releases.

Consider using INetLib only for wireless communication if an application will be delivered exclusively for a 3.x or 4.x device that already supports INetLib.

4.1.3 Network Synchronization

In addition to the synchronization cradle (serial port or USB port), Palm handhelds can also synchronize with desktop computers across a TCP/IP network. Therefore, you can synchronize your Palm handheld using GPRS connectivity to your desktop computer or potentially a server.

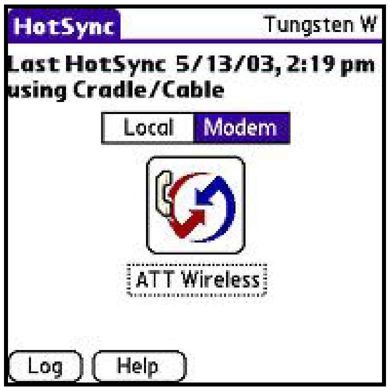
To perform a HotSync operation using GPRS, set the HotSync application to synchronize via ‘Modem’ and provides the Internet name or IP address of the host desktop computer.

The following screen shots in Figure 7 illustrate the configuration on a Tungsten W:

- The first screenshot shows modem (GPRS) synchronization instead of local cradle synchronization.
- The second screenshot shows a direct network (GPRS) connection instead of dialing to another (analog) modem.
- The third screenshot shows the desktop computer’s host name and IP address.

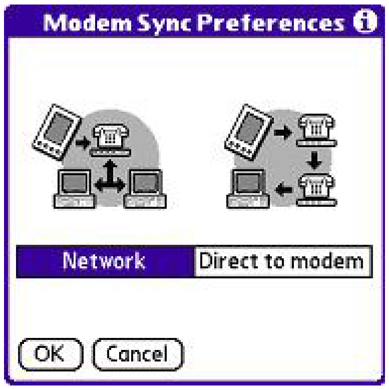
Note: The IP address 64.58.76.226 is provided for this example. The actual address will be the Internet address of your desktop computer.

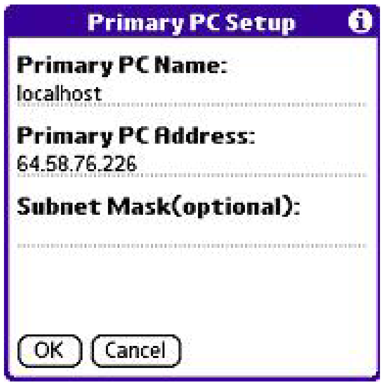
Figure 7 Synchronizing Using GPRS Instead of a Cradle



Last HotSync 5/13/03, 2:19 pm using Cradle/Cable

ATT Wireless





Primary PC Name: localhost

Primary PC Address: 64.58.76.226

Subnet Mask(optional):

When the user taps the HotSync icon, a GPRS network connection is established and networked synchronization proceeds in a fashion similar to cradle synchronization. The following lists the pros and cons of basing applications on this approach.

Pro: Device, Licensee, and Network Independent

Network HotSync has been available since Palm OS 2.0 when NetLib was introduced in 1998. Network HotSync is connection, network and device independent.

Pro: Conduit Synchronizes for Both Wired and Wireless

For Palm applications that already synchronize data via a desktop computer, the Palm HotSync can serve as both the wired and wireless data synchronization solution, and may not require any actual application development.

Instead of using NetLib or INetLib for wireless synchronization and the HotSync conduit for wired synchronization, a HotSync conduit can synchronize for both, simplifying the number of ways used to update information.

Con: Slow Synchronization Via GPRS

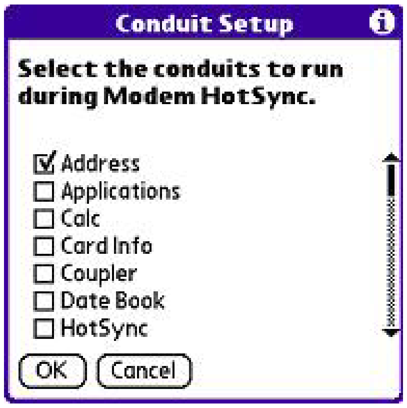
Synchronization using GPRS can be slow. The HotSync application was originally designed to synchronize via the cradle, not necessarily using wireless networks.

For example, synchronizing a Palm Tungsten W to a desktop computer with using GPRS can take around three to ten minutes, depending on the number of applications and conduits installed.

To reduce synchronization time, non-critical application data can be skipped during modem (GPRS) synchronization. Uncheck such applications from the Conduit Setup dialog in the HotSync application.

The screenshot in Figure 8 shows that the data in the Address book application is the only data that will be synchronized via modem (GPRS) synchronization.

Figure 8 Conduit Configuration Screenshot



Select the conduits to run during Modem HotSync.

- [✓] Address
- [ ] Applications
- [ ] Calc
- [ ] Card Info
- [ ] Coupler
- [ ] Date Book
- [ ] HotSync

<u>Con: Difficulty of Reaching Desktop Computer from the Internet</u>

The desktop computer must be accessible from the Internet. Network configurations involving firewalls, port restrictions, and network-address translation can potentially prevent access.

4.1.4 Telephony Management

The Palm OS provides a set of functions in its Telephony API for managing communications-related functions. The Telephony API is available in devices with Palm OS version 4.0 or higher. Palm documents this API in the Palm OS Programmer's API Reference—Telephony Calls. For sections relevant to wireless programming, see the subsections Telephony Basic Services, Telephony Network, Telephony Calls, Telephony SMS, and Telephony Phonebook in http://www.palmos.com/dev/support/docs/palmos/ReferenceTOC.html

The Telephony Calls API manages voice calls by initiating, holding, conferencing, rejecting, and ending calls and by sending DTMF characters. The set can also manage emergency (EMC) calls.

The following functions are a sample of the API from the Telephony Calls subset.

- **TelSpcAcceptCall**—Accept an incoming voice call.
- **TelSpcCallNumber**—Initiate a voice telephone call.
- **TelSpcCloseLine**—End a voice telephone call.
- **TelSpcConference**—Initiate a conference call.

- **TelSpcHoldLine**—Put the current voice line on hold.

A useful function from the Telephony Network section is:

- **TelNwkGetSignalLevel**—Retrieve the selected network carrier signal level.

Other useful functions are available in the Telephony Basics section:

- **TelPowSetPhonePower**—Turn the phone on or off.
- **TelGetCallState**—Retrieve the current telephone call state information.
- **TelInfGetInformation**—Retrieve brand, model, and revision information for the phone.

Handspring developed its own Telephony Management Library for its Treo smart phone line running Palm OS 3.5.2. The functionality in Handspring's library is similar to the functionality provided by the Palm Telephony API. However, the API functions are not the same, and if your application operates on both Palm and Handspring devices, you will have to invoke the appropriate API.

Handspring's API is available at:

http://www.handspring.com/developers/Devkit2/HandspringPhoneLibAPIRef_Win.exe

4.1.5 Short Message Service (SMS)

Short Message Service (SMS), also known as text messaging, enables mobile devices to send and receive short messages, with addressing based on the device's telephone number or Internet e-mail address. Messages contain only alphanumeric characters and are typically about 160 characters in length. Examples of SMS messages include a private conversation between two individuals or news and weather updates.

SMS can also be used in conjunction with GPRS as a means of notifying applications about new information. For example, a mail system could send an SMS message to a mail client to indicate that new mail is available. The mail client could then initiate a GPRS connection to download the e-mail. Such an approach is useful because it does not require the mail client to poll for new e-mail or to maintain a GPRS connection.

You can send or receive SMS messages even during a phone call, as SMS uses the cellular network's control channels, which are always available.

The Palm OS provides a set of functions in its Telephony API called Telephony SMS to read, write, and delete SMS messages in native applications. The Telephony API is available in devices in Palm OS version 4.0 or higher. Palm documents this API in Palm OS Programmer's API Reference—Telephony SMS. See:

http://www.palmos.com/dev/support/docs/palmos/TelephonySMS.html

The following functions are a sample of the Telephony SMS API:

- **TelSmsReadMessage**—Retrieve a delivered message from a storage area.
- **TelSmsSendMessage**—Send an SMS message.
- **TelSmsSelectStorage**—Select a storage area on phone as current SMS storage area.
- **TelSmsGetMessageCount**—Retrieve total number of message slots and number of filled slots for specified message type.

Handspring's SMS API for Palm OS 3.5.2 devices is available at:

http://www.handspring.com/developers/Devkit2/HandspringPhoneLibAPIRef_Win.exe

4.1.6 Subscriber Information Module (SIM)

The Subscriber Information Module (SIM) is a small computer chip in GSM mobile phones that contains individual subscriber information. The subscriber phone number and phone book entries are examples of information stored on a SIM chip. When a subscriber changes phones, the SIM chip can transfer to the new phone to preserve phone book entries, as well as the user identity.

The Palm OS provides a set of functions in its Telephony API called Telephony Phone Book to create, modify and delete phone book entries in a SIM module. The Telephony API is available in devices with version Palm OS 4.0 or higher.

The following functions are examples of the Telephony Phone Book API:

- **TelPhbAddEntry**—Add or replace an entry in currently selected phone book.
- **TelPhbDeleteEntry**—Delete an entry from the currently selected phone book.
- **TelPhbGetEntryCount**—Retrieve total number of entries and filled entries in the currently selected phone book.

Palm documents this API in Palm OS Programmers API Reference—Telephony Phone Book available at: http://www.palmos.com/dev/support/docs/palmos/TelephonyPhonebook.html

Handspring's SIM API for Palm OS 3.5.2 devices is available at:

http://www.handspring.com/developers/Devkit2/HandspringPhoneLibAPIRef_Win.exe

4.2 Web Applications

An alternative to developing a native application is to develop a Web-based application. You will need to consider which browser to use and the suitability of this approach for the particular application.

4.2.1 Third-Party Browsers

Palm does not incorporate a "standard" browser application as part of the operating system read-only memory (ROM.) As a result, several companies have released web browsers for Palm handhelds.

Table 7 identifies the major browsers and their capabilities.

Table 7 Major Browser Capabilities

Browser	OS Version	Supported Web Technologies
AvantGo Publisher: AvantGo	OS 3.0 or later	HTML 3.2. Wireless-capable browser, typically caches web content during wire-line synchronization. Free.
Blazer 2.0 Publisher: Handspring	OS 3.5 or later	HTML 3.2, WML 1.2, cHTML, xHTML, HDML 3.0, 128-bit SSL.

Browser	OS Version	Supported Web Technologies
NetFront Publisher: ACCESS	OS 5.0 or later	HTML 4.01, xHTML, cHTML, WML 1.3, WAP 1.1 and 2.0, SSL 3.0. Packaged with Sony Cliés.
Palm WAP Browser Publisher: Palm	OS 4.1 or later	WML. Packaged with Tungsten T and Tungsten W.
Palm Web Browser Pro 1.0 Publisher: Palm	OS 5.0 or later	HTML, WML, cHTML, xHTML, SSL, JavaScript, CSS. Packaged with Palm Tungsten T.
PalmSource Web Browser 2.0 Publisher: Palm	OS 5.1 or later	HTML 4.01, Javascript 1.5, SSL 3.0. No proxy. Packaged with Palm Tungsten C.

4.2.2 Advantages and Disadvantages of a Web-based Approach

If Palm handhelds are already the target for an existing web application, the application may run well and unchanged on wireless Palm handhelds.

However, many web applications were designed for a desktop screen much larger than a handheld device as well as for higher-speed networks. Framesets, ActiveX, and VBScript are examples of Web technologies not available for Palm browsers.

Instead of trying to accommodate both computing environments in one web application, many web content providers are providing a separate set of pages for handhelds and mobile phones. For example, the content from the following providers has been designed with handhelds in mind.

- Yahoo! http://wap.oa.yahoo.com
- Earthlink http://mobile.earthlink.net

4.2.3 Web Clipping

Web Clipping is a Palm, Inc. technology that combines static HTML and graphics from a web application into a database that is loaded on the device prior to using the application. Web Clipping was introduced in 1999 with the release of Palm OS 3.2 and the Palm VII. The Web Clipping browser uses INetLib to retrieve dynamic content. The advantages and disadvantages of Web clipping are as follows:

Pro: Minimal Wireless Network Access

Because static content such as graphics, search pages, or help files are pre-loaded on the device, the wireless network is only accessed for dynamic content such as news headlines, stock quotes, or an e-mail inbox list. Thus, wireless transmissions are significantly reduced, improving the user experience and reducing wireless charges.

Pro: Reuse of Existing Web Applications

Static content from an existing web application can be incorporated into a Web Clipping application.

Con: Device Updates for Application Updates

While preloading static content improves the user experience, updates to static content must be repackaged into a Web Clipping database and loaded onto the Palm device.

Unlike desktop web applications that always send static content, Web Clipping applications require application updates whenever the static content changes.

Con: Not on All Devices

Web Clipping is unavailable on some devices because the underlying INetLib is not available on some third-party devices and is unavailable on Palm OS 5.x devices. Web clipping is also geared towards a simplified version of HTML 1.0 with little support for client side programming and graphics.

As with INetLib, consider using Web Clipping for wireless communication if an application will be delivered exclusively for a 3.x or 4.x device, where INetLib is available.

5. Development Tools

This section identifies the major software development tools available for developing wireless applications for Palm platforms.

5.1 CodeWarrior for the Palm OS

Metrowerks CodeWarrior for the Palm OS is the most popular commercial C/C++ Palm OS software development tool. Libraries and headers from the Palm OS SDK are compatible with CodeWarrior. In addition, beta versions of the Palm OS SDK are generally made compatible with CodeWarrior before other third-party tools are integrated with the new SDK versions.

Because CodeWarrior is the most widely used development tools for the Palm OS, the example applications in this document have been developed with CodeWarrior.

For more information, see Metrowerks CodeWarrior Development Studio for Palm OS® Platform, Version 9.0 at:

http://www.metrowerks.com/MW/Develop/Desktop/PalmOS/Professional

5.2 PRC-Tools: GCC for the Palm OS

The PRC-Tools package is a collection of tools supporting C and C++ programming for the Palm OS. It includes the GNU Compiler Collection (GCC) in addition to other development tools.

The PRC-Tools package is an attractive choice because it is free. Code developed in PRC-Tools is easily portable to CodeWarrior and vice versa.

For more information, see http://prc-tools.sourceforge.net .

5.3 AppForge MobileVB

The AppForge MobileVisual Basic (VB) commercial development package integrates with Microsoft Visual Basic. Visual Basic developers can leverage their VB knowledge to quickly start developing Palm OS application.

In addition to supporting the Palm OS, AppForge MobileVB has cross-platform support for the Pocket PC and Symbian platforms. AppForge MobileVB provides native controls for interfacing with Palm's Net Library and Internet Libraries.

For more information, see:

http://appforge.com/prod/af_prof/index.html .

5.4 Satellite Forms

The Satellite Forms commercial development package from Pumatech offers its own proprietary language for handheld application development. Using Satellite Forms, you can develop applications for both the Palm OS and Pocket PC platform. Satellite Forms provides a native control that provides access to Net Library functionality.

For more information, see:

http://www.pumatech.com/sf_mad_main.html .

6. Security

For a complete security solution, developers should consider security features provided by GPRS technology, connectivity between the AWS GPRS network and the customer network, the security features and mechanisms available within the Palm OS, and third-party software solutions.

For an additional discussion of security, see Handheld Security for the Mobile Enterprise, Palm Inc., September 2002 at:

http://www.palm.com/pdfs/handheld_security.pdf

6.1 AT&T Wireless GPRS Network

Air-interface ciphering (encryption) in GPRS is similar to encryption in the GSM voice network. The encryption algorithm is called GPRS Encryption Algorithm (GEA).

The GPRS network encrypts data communication between the modem and a portion of the network infrastructure called the Serving GPRS Support Node (SGSN).

Beyond the SGSN, the data is not encrypted. However, it does travel across a private network until it reaches the gateway to external networks called the Gateway GPRS Support Node (GGNS).

Beyond the GGSN, AWS offers a number of options to secure the connection to customer networks. These options include the use of Frame-Relay Permanent Virtual Circuits and VPN connections for Internet-based connection. Refer to the Developer Program for more information.

6.2 SSL in Palm OS 5.0

Palm OS 5 provides an implementation of the Secure Sockets Layer (SSL) protocol at the system level that can be applied to sockets from the Net Library. The SSL 3.0/TLS 1.0 services built into Palm OS 5 enable secure end-to-end connections over the Internet with strong (128-bit) encryption.

In addition, the SSL Library provides data buffering for sockets sending encrypted or “clear text” data.

For an overview of the SSL Library from the Palm OS Programmer’s Companion, see:
http://www.palmos.com/dev/support/docs/palmos/SSLIntro.html

6.3 Security Programming Libraries

Because Palm OS 4.1 and earlier versions do not have native support for cryptography libraries, Palm and third-party vendors have filled the void by developing programming libraries to support digital signatures, authentication, and encryption for native Palm application development.

The Palm Cryptographic Manager from Palm provides cryptographic services for applications running on Palm OS 3.0 or higher. It features strong encryption via AES, HMAC-SHA1 message authentication, and SHA-1 digests. In addition, the Palm Cryptographic Manager is built to comply with FIPS 140-2 Level1.

For pricing and a product datasheet, contact a representative from the Palm Enterprise team at the following URL:

http://www.palm.com/enterprise

In addition to the Palm Cryptographic Manager, the following third-party Palm Solution Providers provide security libraries for the Palm OS:

- **Blueice Research Multipass Client—** http://multipass.com/MultipassClient.pdf
- **Certicom Security Builder Client—** http://www.certicom.com/products/securitybuilder/securitybuilder_feat.html
- **Diversinet Corp Passport—** http://diversinet.com/products/passone.asp
- **Ntru Cryptosystems Inc. Security Toolkit for Palm—** http://www.ntru.com/

6.4 Virtual Private Networking (VPN)

An effective way of securing communications is to employ a Virtual Private Networking Solution. This approach provides end-to-end security, employing encryption, authentication, and access-control mechanisms.

The following is a list of some VPN products from Palm Solution Providers:

- **Certicom MovianVPN**—http://www.certicom.com/
- **Mergic VPN**—http://www.mergic.com/
- **SafeNet SoftRemotePDA**—http://www.safenet-inc.com/

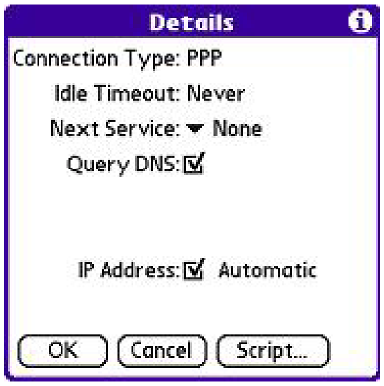
7. Power Management

Palm devices are highly efficient in how they manage power. For example, they will turn off after a certain number of minutes of inactivity. These power events, however, can affect communications-oriented applications. This section discusses the network idle timer, powering off, and low battery situations.

7.1 Network Idle Timer

For most GPRS connections, an idle timer setting, as shown in Figure 9, can be defined in the network configuration to disconnect the connection when no network activity occurs during a specified period of time.

Figure 9 Screenshot Showing Network Idle Timer


Details

Connection Type: PPP

Idle Timeout: Never

Next Service: None

Query DNS: ✓

IP Address: ✓ Automatic

However, as previously noted, no disconnect event is posted into the event queue. Networking applications should use the NetLibOpen, NetLibOpenCount, and NetLibConnectionRefresh functions when making network requests to reestablish the connection if terminated by the idle timer.

7.2 Power Off

When the device power is turned off, the Palm OS terminates the GPRS connection. Power-off situations include the automatic shut-off and the user manually powering off the device using the power button.

All applications are notified of a power-off request by receiving the application launch codes sysNotifySleepRequestEvent and sysNotifySleepNotifyEvent prior to the system turning off power.

However, there may be situations when the device should not be powered off when a GPRS connection is active. An application may want to periodically call the EvtResetAutoOffTimer function to prevent the device from entering sleep mode or complete several send and/or receive operations, or leave the connection open to wait for a data request. When finished with the network communications, it should use the Notification Manager to post these events to allow the device to turn power off.

7.3 Low Battery

The Palm OS automatically checks the power level for battery intensive operations. A standard dialog from the operating system is displayed and gives the user the option to cancel or continue.

In addition, a native application can use the SysBatteryInfo API function to check the battery status to avoid creating a connection or starting a long network operation on low power.

7.4 Monitoring Battery Status

A communications-oriented application may choose to not initiate certain transactions if the battery level is low, or to make other decisions, based on the battery level. The **SysBatteryInfo** function in the System Manager section of the Palm OS Programmer's Reference returns battery information.

8. Other Programming Considerations and Approaches

This section discusses additional considerations, including blocking versus non-blocking sockets, testing strategies, an approach that combines HotSync with TCP/IP communications, applications that use application servers, and ROM versions.

8.1 Net Library: Blocking versus Non-blocking Sockets

The Net Library supports both blocking and non-blocking sockets. In a multi-threaded operating system, you can create a separate thread for socket communications. While this networking thread processes socket events, threads that process other events such as user interface events can continue processing without waiting for the socket events.

However, the Palm OS has a single thread for all user interface processes. When blocking sockets wait for socket events, user interface events are not processed. The user can tap on the screen, push a hardware button, and turn the device on and off in vain. The device will not respond until a timeout event is received from the blocking socket.

Fortunately, the Net Library supports non-blocking sockets. When you create a non-blocking socket, future calls return a result code immediately for the current status of the operation instead of waiting until the timeout. You can call the same Net Library API numerous times until the operation is completed instead of calling the Net Library API once for its completion or waiting for a timeout to occur.

The major advantage of using non-blocking sockets is the opportunity to handle user interface events during socket operations. The user interface can update the progress with a timer or graphical indicator, showing that the device is active. The user can control the network operation by clicking a cancel button and receiving an immediate response to the request.

However, the main disadvantage of using non-blocking sockets is more complex code. The application must retry API calls numerous times to check the progress, instead of just once with a blocking socket. The application must manage partial buffers when sending and receiving data instead of having blocking sockets handle these tasks.

A detailed explanation of blocking and non-blocking sockets is available from Greg Winton’s book “Palm OS Network Programming.”

8.2 Testing Strategies

To test your application, it is generally most effective to begin with emulators and simulators, and then to progress to using real networks.

8.2.1 Desktop with Palm OS Emulator and Simulator

In addition to developing non-networking Palm OS applications, the Palm OS Emulator and Palm OS Simulator are invaluable debugging and testing tools.

Both the Emulator and Simulator can route Net Library and Internet Library requests to the TCP/IP stack of the host computer. Here are some tips:

- **Palm OS Emulator**: Confirm that the “Redirect NetLib calls to host TCP/IP” is checked in the Emulator Properties dialog.
- **Palm OS Simulator**: Confirm that the “Redirect NetLib Calls to Host TCP/IP” menu option is checked under the Communications menu under the Settings menu.

The core application can be easily debugged and tested using the reliable wired network connection of the desktop computer.

8.2.2 Device with Short-range Networks

Once the application performs well on the desktop, consider testing the application on a device using a short-range network technology such as 802.11b or Bluetooth. These networks are readily available for use with Palm devices, and testing with them enables useful testing without incurring GPRS airtime costs.

For 802.11b, the Palm Tungsten C has a built-in 802.11b radio that can communicate with any 802.11b access point. In addition, Intel offers an attachable 802.11b sled for the Palm m500 series.

For Bluetooth, some desktop computer manufacturers offer built-in Bluetooth. Numerous manufacturers offer stand-alone Bluetooth access points and dongles that connect to desktop computer.

8.2.3 Device with GPRS Network

Avoid the temptation to stop testing after the emulator and indoor network solution testing is complete. Real-world testing of network applications with GPRS-enabled devices is essential.

Test intermittent coverage situations. Test in areas where coverage is poor or unavailable. Start a network connection where coverage is adequate and move to poor coverage areas during a transaction. Make sure the application handles loss of network coverage properly and provides appropriate error messages.

Test lost connection situations. Create a GPRS connection and initiate a networking transaction. When the transaction is complete, power off the device. Powering off the device automatically closes the GPRS connection. Power on the device and confirm that you can initiate another networking transaction. Confirm that the application properly reestablishes the GPRS connection.

In addition to GPRS network testing, you may want to test with devices on other wireless networks such as Mobitex and CDMA. Applications that use Net Library and Network HotSync technologies should work on these other networks as well. However, differences in network performance, IP address management and proxy servers may provide differing application results.

8.3 Hybrid Network HotSync and TCP/IP Solution

Consider using a hybrid Network HotSync and TCP/IP based networking solution when a portion of data requires real-time updates and the rest requires infrequent but volume-intensive updates, such as on a daily or weekly basis.

For example, a stock trading application for a broker may need a traditional HotSync conduit to load a device with detailed customer account information. However, current stock quotes require fast, real-time response that Network HotSync operations do not offer. To update individual quote information, the application can use the Net Library to update just the stock quotes.

8.4 Using an Application-Server

An alternative to developing an application for the Palm environment is to make it available via an application server. In this approach, which is sometimes referred to as “remote control,” the application operates on a server at a central location or on a user’s desktop. This can be done with products such as Virtual Networking Computing (VNC), available from various vendors for the Palm platform, and Win-Hand (http://www.win-hand.com/). These are remote display systems that allow users to view a remote computing desktop environment on their Palm device. The server sends screen updates to the client, and the client sends user input to the server. This approach is feasible using GPRS connections. VNC server software is available for a wide variety of platforms, including Linux, Macintosh, Microsoft Windows, and Unix.

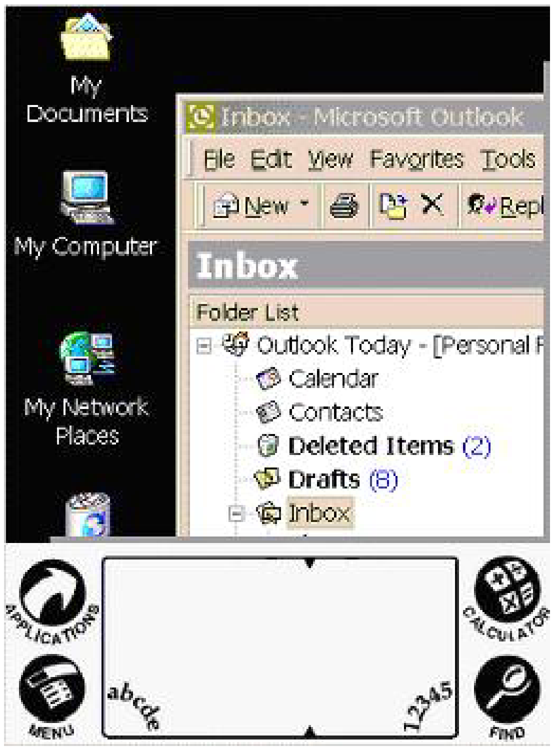
The advantage of this approach is that applications can be maintained at a central location, and no software needs to be distributed for the mobile devices. One item to consider, however, is that many existing applications assume a screen size larger than what is currently offered by the Palm device. To avoid inconvenient screen scrolling by the user, it may be best to reformat the display output to the parameters of the Palm screen.

With an application server and a GPRS-capable Palm handheld, the Palm handheld can view the desktop, launch applications, terminate process, and even restart the remote computer.

An application server is ideal for system and network administrators to remotely manage machines using the Palm handheld. Administrators can perform urgent tasks when away from a desktop computer that is equipped with the remote access software. In addition, you can use application servers in conjunction with Virtual Private Networking products to secure communications.

However, an application server may not be an ideal solution for the typical professional. The remote (e.g., desktop) computer must be configured to reside directly on the Internet. The user must know the host computer's network address and ensure that it does not change (such as through DHCP). Finally, screen updates can be slow, typically taking two to five seconds, depending on the screen activity. The screen image in Figure 10 shows a remote Windows desktop viewed on a Palm device.

Figure 10 Screenshot of Remote Desktop on a VNC Client



8.5 ROM Version

Different read-only-memory (ROM) versions of the Palm OS can have different implications for wireless applications. An application may want to check that the appropriate ROM version is installed in the device.

The application should invoke the **System Version Feature** in the Palm System Support section of the Palm OS Programmer's Reference. See: http://www.palmos.com/dev/support/docs/palmos/SystemFeatures.html#1013505 .

Use the following parameter sequence with the FtrGet (Feature Get) API function to return the ROM version:

```
FtrGet(sysFtrCreator, sysFtrNumROMVersion, &romVersion);
```

9. Sample Applications

This section provides examples of GPRS application development for the Palm platform. It begins with a discussion of the development environment, and then demonstrates two applications. One involves a simple Hypertext Transport Protocol (HTTP) application, and the other builds on this to retrieve stock quotes. These applications, including source and object code, are available from the Palm section of the Data Developer Web site:

http://www.attws.com/developer/technologies/palm/

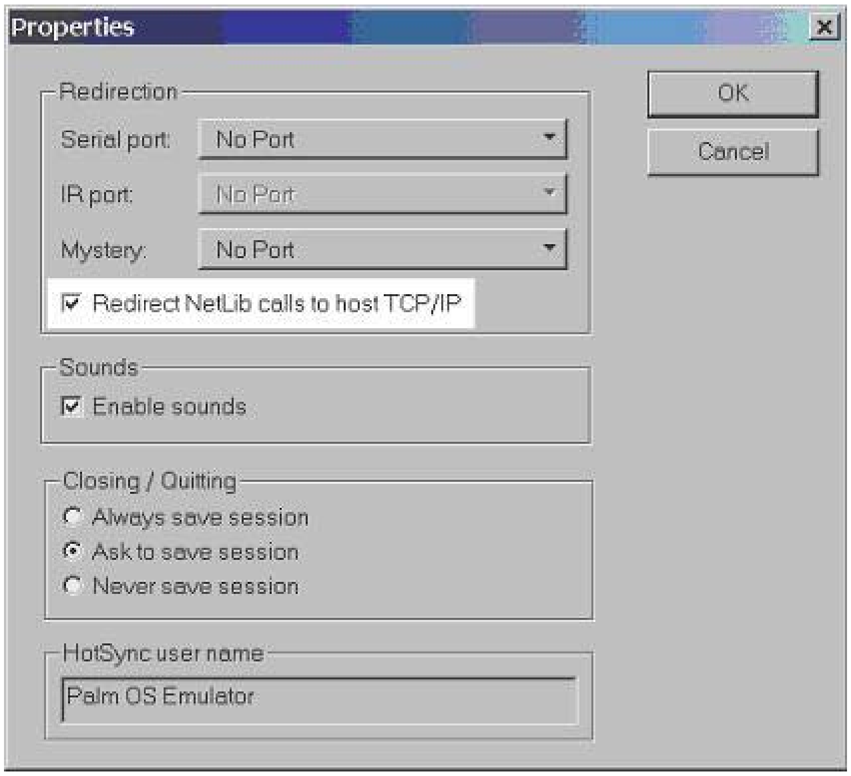
9.1 Development Environment

You need the following development environment to compile the source code for both sample applications.

- Windows 98/NT/2000/XP.
- Metrowerks CodeWarrior for the Palm OS, version 7.0 or later.
- Palm OS Emulator, which is installed with Metrowerks CodeWarrior or can be downloaded from the PalmSource website.

Note: Confirm that the “Redirect NetLib calls to host TCP/IP” box is checked in the Emulator Properties dialog, as shown in Figure 11. When this is not checked, the emulator tries to use the Palm OS Connection and Network settings to establish a connection instead of using the host computer’s Internet connection.

Figure 11 Screenshot Showing NetLib Calls Redirection



Redirection

Serial port: No Port

IR port: No Port

Mystery: No Port

[✓] Redirect NetLib calls to host TCP/IP

Sounds

[✓] Enable sounds

Closing / Quitting

( ) Always save session

(●) Ask to save session

( ) Never save session

HotSync user name

Palm OS Emulator

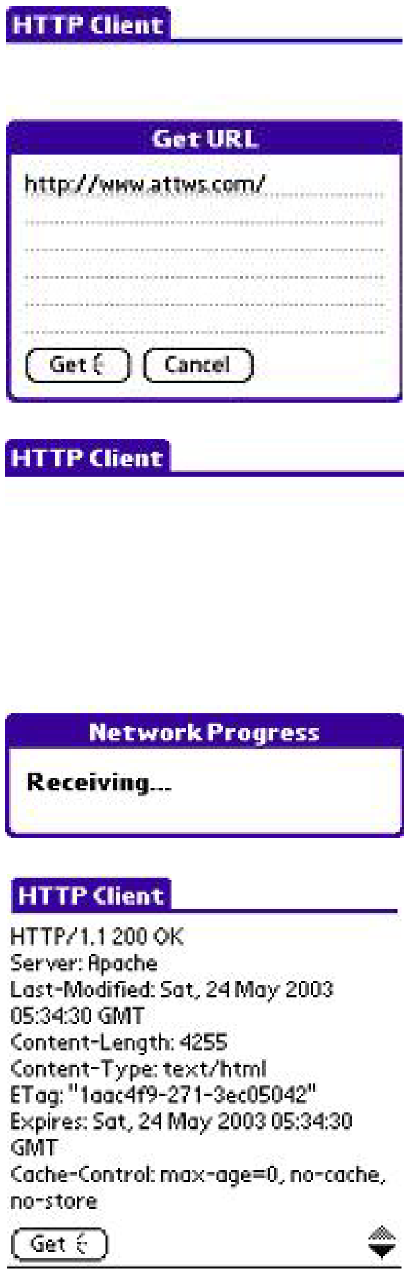
9.2 Sample Application: HTTP Client

The HTTP client application retrieves web content from a URL provided by the user.

9.2.1 User Interface

To retrieve the contents of a URL, a user taps the “Get” button from the main form. A modal form is displayed with a field for entering the URL. Users can enter a standard URL including the http:// prefix or start with the hostname and omit the http:// prefix. See the screenshots shown in Figure 12.

Figure 12 Screenshots of Sample Application User Interface



http://www.attws.com/

```
HTTP/1.1 200 OK
Server: Apache
Last-Modified: Sat, 24 May 2003
05:34:30 GMT
Content-Length: 4255
Content-Type: text/html
ETag: "1aac4f9-271-3ec05042"
Expires: Sat, 24 May 2003 05:34:30
GMT
Cache-Control: max-age=0, no-cache,
no-store
```

After entering the URL, a user taps the “Get” button in the modal form to start the wireless transaction. Note in the screenshot above that this button contains a small symbol to imply that a network or wireless communication starts when the user selects the button. Palm has adopted a standard character to indicate that a network communication will be initiated when the user taps the button. For unsecured HTTP communications you would postfix the button's label with a 0x15 in standard Palm font using Code Warrior's resource constructor. For secured HTTPS communications, the character 0x14 should be used instead.

When the user taps the "Get" button, the application displays a progress dialog and begins wireless communications. When the application receives a response, it copies the response into a large field in the main form.

For this sample application, the application requests the first 1024 bytes received in the HTTP response. In addition, the application uses a 15 second timeout for DNS lookups and send and receive operations.

9.2.2 Source Code: HTTP Class

The Net Library calls for this sample application are wrapped into a C++ class called HTTP. The following sections identify the Net Library calls performed in each method in the HTTP class.

9.2.2.1 Method init

This method confirms that the Net Library is available and retrieves its library reference number for future Net Library API calls. The SysLibFind API function returns the Net Library reference number.

Note: Net Library calls are denoted in bold.

```
void HTTP::init(void)
{
   //  find the library reference number of the Net Library

   if (SysLibFind("Net.lib", &m_wLibRef))
      throw sysErrLibNotFound;
}
```

9.2.2.2 Method open

This method opens the Net Library. Opening the Net Library initializes the library and starts all attached network interfaces. The primary API function to open the Net Library is the NetLibOpen call.

It is possible, however, that the Net Library is open but the GPRS connection has been closed due to the network connection idle timer expiring or the device being turned off by the user.

Therefore, this method uses NetLibOpenCount to determine if the Net Library is already open. If the Net Library is already open, NetLibConnectionRefresh restarts connections that have been closed. Otherwise, NetLibOpen starts the Net Library and connects to the network.

```
void HTTP::open(UInt16& a_wInterfaceError)
{
   //  confirm the Net Library is available (from init method call)

   if (!m_wLibRef)
      throw sysErrLibNotFound;

   //  open the Net Library to establish a GPRS connection

   Err err;
   UInt16 wOpenCount;

   NetLibOpenCount(
      m_wLibRef,
      &wOpenCount);

   if (wOpenCount)
   {
      // Net Library is open... verify interface is actually up

      UInt8 byAllInterfacesUp;

      err = NetLibConnectionRefresh(
         m_wLibRef,
         true,
         &byAllInterfacesUp,
         &a_wInterfaceError);

      if (err)
         throw err;
   }
   else
   {
      err = NetLibOpen(
         m_wLibRef, number
         &a_wInterfaceError);

      if (err == netErrAlreadyOpen)
         return;  //  no error... connection already up

      if (err)
         throw err;
   }
}
```

9.2.2.3 Method lookup

This method resolves the host name in the URL to its IP address using the NetLibGetHostByName API function.

Note: A private method called parseHostFromURL extracts the host name from the URL. The URL and the host name are stored in the class.

In addition, note that up to three IP addresses can be returned for a host name. In this example, the first IP address returned is used.

```
void HTTP::lookup(const Char* a_pcURL)
{
   //  confirm the Net Library is available (from init method call)

   if (!m_wLibRef)
      throw sysErrLibNotFound;

   //  confirm URL string is not empty or NULL

   if (!a_pcURL)
      throw netErrDNSBadName;

   //  parse the host name from the URL

   if (!parseHostFromURL(a_pcURL))
      throw netErrDNSBadName;

   const Char* pcHost = (const Char *)MemHandleLock(m_hHost);

   //  perform the DNS lookup

   Err err;
   NetHostInfoBufType hostInfo;
   NetHostInfoPtr pHostInfo;

   pHostInfo = NetLibGetHostByName(
      m_wLibRef,
      pcHost
      &hostInfo,
      m_lTimeoutTicks,
      &err);

   MemHandleUnlock(m_hHost);

   if (err)
   {
      MemHandleFree(m_hHost);
      MemHandleFree(m_hURL);

      m_hHost = 0;
      m_hURL = 0;

      throw err;
   }

   //  get pointer to first address (using first)

   NetIPAddr* pAddress = (NetIPAddr *)(pHostInfo->addrListP[0]);
   m_ipAddress = (NetIPAddr)*pAddress;
}
```

9.2.2.4 Method connect

This method creates a socket and attempts to connect the host to the socket. The NetLibSocketOpen API function creates the socket and the NetLibSocketConnect API function connects the socket to the host.

```
void HTTP::connect(void)
{
   //  confirm the Net Library is available (from init method call)

   if (!m_wLibRef)
      throw sysErrLibNotFound;

   //  confirm the host name was resolved to an IP address

   if (!m_ipAddress)
      throw netErrDNSBadName;

   //  create a new socket

   Err err;

   m_sockRef = NetLibSocketOpen(
      m_wLibRef,
      netSocketAddrINET,
      netSocketTypeStream,
      netSocketProtoIPTCP,
      m_lTimeoutTicks,
      &err);

   if (err)
      throw err;

   //  connect the socket to the destination

   NetSocketAddrINType socketAddr;

   MemSet(&socketAddr, sizeof(socketAddr), 0x00);

   socketAddr.family = netSocketAddrINET;
   socketAddr.port = 80;
   socketAddr.addr = m_ipAddress;

   Int16 iResult = NetLibSocketConnect(
      m_wLibRef,
      m_sockRef,
      (NetSocketAddrType*)&socketAddr,
      sizeof(socketAddr),
      m_lTimeoutTicks
      &err);

   if (iResult)
      throw err;
}
```

9.2.2.5 Method send

This method creates the HTTP request and sends the request to the host connected to the socket from the send method. The NetLibSend API function sends the request.

```
void HTTP::send()
{
   //  confirm the Net Library is available (from init method call)

   if (!m_wLibRef)
      throw sysErrLibNotFound;

   //  confirm the host name was resolved to an IP address

   if (!m_ipAddress)
      throw netErrDNSBadName;

   //  confirm that socket is open

   if (!m_sockRef)
      throw netErrSocketNotOpen;

   //  send the data

   Err err;

   Char *pcURL = (Char *)MemHandleLock(m_hURL);
   Char *pcHost = (Char *)MemHandleLock(m_hHost);

   MemHandle hSend = MemHandleNew(512);
   Char* pcSend = (Char *)MemHandleLock(hSend);

   StrCopy(pcSend, "GET ");
   StrCat(pcSend, pcURL);
   StrCat(pcSend, " HTTP/1.1\r\n");
   StrCat(pcSend, "Cache-Control: no-cache\r\n");
   StrCat(pcSend, "Host: ");
   StrCat(pcSend, pcHost);
   StrCat(pcSend, "\r\n");
   StrCat(pcSend, "Accept: */*\r\n");
   StrCat(pcSend, "Connection: close\r\n\r\n");

   Int16 iSent = NetLibSend(
      m_wLibRef,
      m_sockRef,
      (void *)pcSend,
      StrLen(pcSend) + 1,
      0,
      NULL,
      0,
      m_lTimeoutTicks,
      &err);

   MemHandleUnlock(m_hHost);
   MemHandleUnlock(m_hURL);

   MemHandleUnlock(hSend);
   MemHandleFree(hSend);

   if (err)
      throw err;
}
```

9.2.2.6 Method receive

This method waits for a response to the HTTP request. The NetLibReceive API function waits for the response.

Notice that the /r/n pairs are substituted for /n pairs because the field controls display an unrecognized character for \r.

For this example, the operating system blocks while waiting for the response. In this example, the timeout value is 15 seconds.

```
MemHandle HTTP::receive(void)
{
   //  confirm the Net Library is available (from init method call)

   if (!m_wLibRef)
      throw sysErrLibNotFound;

   //  confirm the host name was resolved to an IP address

   if (!m_ipAddress)
      throw netErrDNSBadName;

   //  confirm that socket is open

   if (!m_sockRef)
      throw netErrSocketNotOpen;

   //  receive the data

   Err err;
   MemHandle hData;

   hData = MemHandleNew(m_wMaxBytes);
   Char* pcData = (Char *)MemHandleLock(hData);

   MemSet(pcData, m_wMaxBytes, 0x00);

   Int16 iRead = NetLibReceive(
      m_wLibRef
      m_sockRef
      (void *)pcData
      m_wMaxBytes – 1,
      0,
      NULL,
      0,
      m_lTimeoutTicks,
      &err);

   if (err)
   {
      MemHandleUnlock(hData);
      MemHandleFree(hData);

      throw err;
   }

   //  substitute /r/n for /n for Palm OS fields to look better

   Char* pcSrc = pcData;
   Char* pcDest = pcData;

   for(; *pcSrc; pcSrc++, pcDest++)
   {
      if (*pcSrc == '\r')
      {
         *pcDest = '\n';

         if (*(pcSrc+1) == '\n')
            pcSrc++;
      }
      else
         *pcDest = *pcSrc;
   }

   *pcDest = '\0';

   MemHandleUnlock(hData);

   return hData;
}
```

9.2.2.7 Method disconnect

This method disconnects from the host and closes the socket created by the open method using the NetLibSocketClose API function.

```
void HTTP::disconnect(void)
{
   //  confirm the Net Library is available (from init method call)

   if (!m_wLibRef)
      throw sysErrLibNotFound;

   //  confirm the host name was resolved to an IP address

   if (!m_ipAddress)
      throw netErrDNSBadName;

   //  confirm socket was opened and connected

   if (!m_sockRef)
      throw netErrSocketNotOpen;

   //  close the socket

   Err err;

   Int16 iResult = NetLibSocketClose(
      m_wLibRef,
      m_sockRef,
      m_lTimeoutTicks,
      &err);

   MemHandleFree(m_hURL);
   MemHandleFree(m_hHost);

   m_ipAddress = 0;
   m_sockRef = 0;
   m_hURL = 0;
   m_hHost = 0;

   if (err)
      throw err;
}
```

9.2.2.8 Method close

This method closes the Net Library using the NetLibClose API function.

The second parameter in the NetLibClose function determines whether the network connection is closed immediately or delayed. By setting this value to false, the idle timer in the network configuration dialog is used to determine when to the close the application.

The delayed setting is recommended. After viewing a URL, the user might switch to an e-mail application to view mail. Instead of reestablishing the GPRS connection, the user can begin network communications immediately in the e-mail application.

```
void HTTP::close(Boolean a_bImmediate)
{
   //  confirm the Net Library is available (from init method call)

   if (!m_wLibRef)
      throw sysErrLibNotFound;

   //  close Net Library to release GPRS connection

   Err err = NetLibClose(m_wLibRef, a_bImmediate);

   if (err == netErrStillOpen)
      return;  //  no error... connection in use by another

   if (err)
      throw err;
}
```

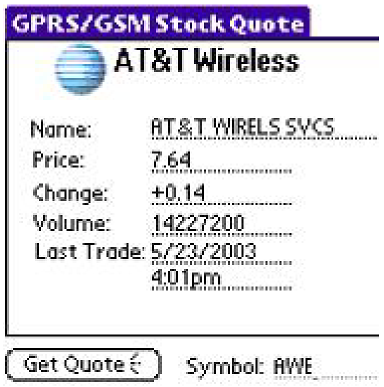
9.3 Sample Application: Stock Quote

The Stock Quote application builds on the HTTP class to display stock quote information using the Yahoo! Finance stock quote service.

9.3.1 User Interface

To retrieve a quote for a stock, a user enters the stock symbol in the Symbol field, as shown in Figure 13.

Figure 13 User Interface for Stock Quote Sample Application



Name: AT&T WIRELS SVCS
Price: 7.64
Change: +0.14
Volume: 14227200
Last Trade: 5/23/2003 4:01pm

Symbol: AWE

After the user enters the stock symbol, the application forms a URL and sends it to the Comma-Separated-Value quote service to retrieve the quote.

9.3.2 Source Code: formmain.cpp

The Yahoo Finance URL is built inserting the stock symbol between the following URL segments.

```
#define URL_PREFIX “http://finance.yahoo.com/d ?s=”
#define URL_SUFFIX "&f=sl1d1t1c1ohgvn"
```

The following code in the HandleControlSelectEvent function builds this URL in response to the user tapping the “Get Quote” button.

```
   g_hURL = MemHandleNew(
      StrLen(URL_PREFIX) +
      StrLen(pcSymbol) +
      StrLen(URL_SUFFIX) + 1);

   Char* pcURL = (Char *)MemHandleLock(g_hURL);
```

```
    StrCopy(pcURL, URL_PREFIX);
    StrCat(pcURL, pcSymbol);
    StrCat(pcURL, URL_SUFFIX);

    MemHandleUnlock(g_hURL);
```

For example, this quote builds the following URL of a quote request for AT&T Wireless (Symbol: AWE):

http://finance.yahoo.com/d?s=AWE&f=sl1d1t1c1ohgvn

The response is a comma-separated list of values with the quote. The Parser class parses the comma-separated values from the response of a quote request for display in fields in the main form.

```
"AWE",16.47,"5/12/2003","4:01pm",-0.04,16.63,16.64,16.17,6300800,
“AT&T WIRELS SVCS”
```

Refer to the actual complete source code available at the AT&T Wireless Developers site for more details.