ProGear™

Technical Reference Guide

Version 1.1

frontpath

frontpath, Incorporated
2841 Mission College Blvd.
Santa Clara, CA. 95054
408-588-8800
www.frontpath.com

Table Of Contents

ProGear Technical Reference Guide

1. Introduction

ProGear is frontpath's premier Information Appliance targeted to the vertical market segments. ProGear couples hardware and software to deliver one of the first ever wireless, truly portable, untethered, broadband-based products capable of enabling vertical partners to fully customize content. Because ProGear is based on the Linux open operating system and x86-compatible processor, value-added resellers (VARs) and developers can customize the device to meet the needs of individual vertical markets.

ProGear provides a standard user interface consisting of the Netscape web browser and a compliment of commonly used browser plug-ins. User interaction is via a touch screen and scroll button. A virtual keyboard and handwriting recognition are available for text entry. ProGear is typically configured with an internal wireless networking card. While in range of a wireless access point, a user simply enters an application's URL and is off and running. ProGear is built on the Linux operating system, with all the power and flexibility associated with a Unix environment. When used as a thin client, ProGear can be supplied with a compact down run-time environment loaded onto its Flash RAM drive. ProGear can also be supplied with a hard disk and full Linux environment.

Document Overview

This document provides development information for VARs and other application developers. The development package contains this document and a collection of reference documents that provide information specific to developing applications, writing drivers, and porting existing applications to ProGear.

This document assumes an understanding of developing web-based applications. Sections that discuss developing software to run on ProGear assume an understanding of developing for the Unix environment and a familiarity with Linux.

ProGear supports the following three standard distributions. See Section 6 for details on developing for the different distributions.

- ”DevGear” development distribution contains a full Linux distribution, as well as the gnu development tools and source code for much of the ProGear software.
- ”ProGear” distribution is a stripped down set of files intended only to support wireless Web access.
- ”ProGear-TC” (thin client) distribution is small enough to be loaded into compact flash memory in a diskless configuration. This version is essentially the same as ProGear except that some files are removed.

This document is outlined as follows:

Section 2 - provides information for developers of server-based applications, including: ProGear's screen size, browser and plug-in specifications, configuring ProGear's default URL, installing additional browser plug-ins, and design guidelines for optimizing applications to ProGear's user interface.

Section 3 - discusses developing software that runs on the ProGear unit, including web applications, browser plug-ins, Linux applications, and device drivers.

Section 4 – provides information on the Remote Software Update feature.

Section 5 - contains hardware specifications and should be of particular interest to those writing device drivers or applications that directly use ProGear's various input/output devices.

Section 6 - contains a detailed description of ProGear's software environment.

Appendices A, B, C, and D - contains copies of setup, user, and support documents for the ProGear; instructions for accessing the ProGear development environment; sample script files; release notes, errata, and answers to frequently asked questions; known bugs and bug reporting/tracking; references to vendor datasheets for key ProGear components; and a bibliography of useful references. In addition, a ProGear developer's source code CD-ROM is available to accompany this document as part of the ProGear SDK.

Note—Software version numbers were accurate at the time this document was released. Please refer to the software files on your disk for current software version numbers.

1.1. Developing for ProGear

For many applications, ProGear serves solely as a client—with applications running on a separate server. Development is the same as for any other web-based application, and only minimally involves the ProGear. Developers may wish to tailor user interfaces to ProGear's screen size, standard set of browser plug-ins, and touch screen interface. Also, developers may wish to tailor the ProGear—for example, by setting its default URL and adding additional plug-ins.

ProGear is built on the Linux operating system, and when needed, can provide all the power and flexibility associated with a Unix environment. For example, when an application must operate while out of wireless range, HTML and media files can be loaded onto the ProGear for local access. In addition, various server functions can be executed locally on the ProGear, such as an onboard web server, scripting languages (e.g., Perl), a Java virtual machine, compiled code, or a database engine. Finally, developers can write onboard applications that run directly under Linux. Developers may also be called on to interface new hardware devices and write associated device drivers.

1.2. Development Environment

ProGear is normally configured with an internal wireless networking card, permanently installed in the unit's PC Card, and a single USB port. Other configurations are available that may be useful during developing:

- A standard unit with an accessible PC Card slot, allowing the wireless networking card to be replaced with other PC Cards.
- A unit containing the ”Development Distribution” of the ProGear environment, complete with tools such as gnu tools, header files for ProGear libraries, Makefiles, and external access to the unit's hard drive.

In most cases, application development and execution occurs on machines other than the ProGear. When data or program files reside on the ProGear, it usually makes the most sense to develop in your normal development environment and then transfer files to the ProGear for test and deployment. The process is simple:

- Connect a ProGear unit to your network (via a PC Card wireless or Ethernet networking card).
- Plug in a keyboard.
- Escape to Linux (ctrl-alt-F2) and login (see errata sheet for usernames and passwords).
- Type "ifconfig" to find the units network address.
- Use FTP to transfer files, or use NFS to remote mount disks.
- Use X-Windows to access the test unit from your development machine.

In those special cases where code is dependent on the ProGear's software environment (e.g., device drivers or C applications that make use of shared libraries), we recommend building software on the ProGear.

2. Developing Server-Based Software for Access via ProGear

ProGear provides an out-of-the-box capability to access web-based applications (assuming that the unit is within range of a wireless access point).

Section 2.1 - describes the characteristics of the standard ProGear web client, including the screen size, browser, standard plug-ins, and suggested design guidelines (e.g., what the interface can and cannot support).

Section 2.2 - describes customizing the ProGear to better support specific web-based applications, including setting the ProGear's default URL, installing additional plug-ins, and installing an alternate web browser. Be sure to review the release notes in the appendix for updates and errata to the information provided below.

2.1. The Standard ProGear Environment

On power-up, a standard ProGear unit invokes a Netscape web browser. The touch screen and stylus provide the equivalent of a mouse. A virtual keyboard and handwriting recognition are used for text input. Please note that there is no way to disable or remove the handwriting recognition feature without corrupting the operating system.

2.1.1. Screen Geometry

The ProGear unit can operate in either portrait or landscape modes. Figures 2.1.1. -1 and 2.1.1. -2, respectively, detail the screen geometry in each mode. Note that when the virtual keyboard or handwriting recognition is invoked, the browser window is automatically resized.

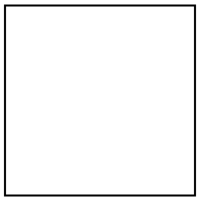


Figure 2.2.1. -1 Portrait Mode Screen Geometry (768x1024)

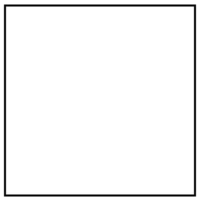


Figure 2.2.1. -2 Landscape Mode Screen Geometry (1024x768)

2.1.2. Browser Specifications

ProGear ships with Netscape Navigator for Linux (ver. 4.7x or newer), which includes built-in support for JavaScript (v. 1.3, ECMA-252 compliant) and Java (JDK v. 1.1). Detailed specifications can be found on Netscape's web site home.netscape.com and technical support can be found at help.netscape.com.

Note—Netscape for Linux does not support functionality specific to the Microsoft Windows environment, such as ActiveX controls.

2.1.3. Browser Plug-Ins

ProGear ships with the following browser plug-ins installed:

- Macromedia Flash 4.0 r1.2
- Adobe Acrobat PDF Viewer 4.0.5

2.1.4. Browser Configuration

The browser is configured to use the directory **/home/webzine** as its working directory. Configuration and other browser support files are stored in **/home/webzine/.netscape**.

2.1.5. Java Virtual Machine

ProGear provides the following for implementing the Java virtual machine:

- Standard Netscape
- Java Specifications

Note—Other Java virtual machines can be included to allow greater functionality.

Netscape supports Java 1.1. The Blackdown group has ported Java 1.2 and 1.3 to Linux and made a Netscape plug-in available. The Java Plug-in requires that webpage authors make changes to their existing HTML code to have JDK 1.1-based applets run using the Java plug-in, rather than Netscape's Java runtime. To make this process easy for webpage authors, Sun Microsystems provides the Java Plug-in HTML Converter, free of charge, to automate this process. The Java plug-in HTML Converter can be downloaded from the HTML Converter download page. For the HTML specification, see the Java Plug-in HTML Specification. In addition, Sun has provided a detailed specification outlining the HTML changes required to use the Java Plug-in, and how webpage authors can implement them "by hand." Documentation on the Java Plugin from Sun can be found at http://www.javasoft.com/products/plugin/1.1.1/docs/index.html.

Please reference the Blackdown organization at http://www.blackdown.org for current information regarding updated Java ports to Linux. The Java ports and their current status can be found at the Java-Linux Ports page at http://www.blackdown.org/java-linux/ports.html. See http://www.blackdown.org/java-linux/docs/support/faq-release/FAQ-java-linux-4.html#browser-plugin for additional information on the Blackdown Java Browser Plugin.

2.2. Customizing the ProGear Web Browsing Environment

Even if your application(s) run on a server, you may wish to make some simple customizations to the ProGear environment. The most common are detailed here.

2.2.1. Run-Time Environment

Upon power-up, the ProGear initializes basic system functions, then su's to user "webzine," starts X-windows, starts a routine to read the scroll button, starts Netscape, and then creates a Menubar. The Menubar provides access to the virtual keyboard and handwriting window, as well as access to the audio volume control and other system settings. Details of the run-time environment are provided in Section 6.

2.2.2. Setting the Default URL

On startup, Netscape will access a pre-defined default web page. The default page is stored in the file **/home/webzine/.netscape/preferences.js** and is initially set to www.frontpath.com.

At that point, the default URL can be set through Netscape's Edit>Preferences dialog. For OEM applications, a more practical approach is to install a modified version of preferences.js, containing the desired URL. See Section 2.2.5 for approaches to distributing such a modification.

2.2.3. Installing Additional Browser Plug-Ins

Individual users cannot download and install browser plug-ins. For OEM applications, you must follow the steps below:

1) Start with a ProGear configured with your baseline software distribution.

2) Install any additional plug-ins that you desire. Please note that you must type in the appropriate Linux command to install plug-ins after downloading.
3) Identify all modified files and directories—what gets modified when a new plug-in is installed, or how can one identify what's changed.
4) Ship a distribution that incorporates the modified files and directories (see Section 2.2.5. for approaches to distributing the modifications).

Note—Please refer to the README file or other documentation that comes with the plug-in software. There may be other libraries or components needed.

2.2.4. Installing an Alternate Browser

frontpath supports the Netscape browser. If you are porting an existing application that has been tailored to an alternate browser, you may wish to install a different browser. This will require significant modification to the ProGear environment. Please contact ProGear developer support at devInfo@frontpath.com.

2.2.5. Distributing Modifications to the Standard ProGear Environment

During manufacturing, frontpath initializes ProGear's flash memory and/or disk drive, and then seals the unit. Thus, it is impractical for an OEM to self-install modified software in the form of mass-produced flash memory or disk drives.

Several techniques are available for installing modifications in an efficient manner:

- Install applications, files, and/or data using the Remote Software Update software. Developers can create their own Remote Update Server or contract with frontpath to provide this service.

- Use a wired or wireless Ethernet LAN to manually download files from your server and then manually install the files onto the ProGear.

- Contract with frontpath to load a custom image (including the developer's software modifications) onto the ProGear's hard drive or Flash RAM drive. These custom drives will be installed into units at the ProGear manufacturing facility. Volume considerations and additional changes will apply.

3. Developing Software to Run on ProGear

As a full Linux environment, ProGear is capable of much more than simply serving as a web browser.

Section 3.1 - provides guidelines for developing and installing software on the ProGear, such as what directories to use.

Section 3.2 - discusses hosting web applications and their associated server(s) on the ProGear.

Section 3.3 - discusses developing applications to run directly under Linux.

Section 3.4 - discusses developing browser plug-ins for the ProGear environment.

Section 3.5 - discusses interfacing hardware devices (e.g., PC Cards) to the ProGear and writing the associated devices.

Section 3.6 – discusses tailoring a ProGear software distribution that incorporates your software.

See Sections 5 and 6 for detailed hardware and software reference materials and the appendices for user interface design guidelines, script files, and instructions on accessing ProGear's development environment.

3.1. Standards for ProGear Development

The following standards are used internally for developing ProGear software. To insure that your software can be distributed without conflicting with software developed by frontpath and/or by other developers, we recommend that you follow these standards for your own developments. The philosophy applied here is adherence to Unix/Linux standards as much as possible while enabling efficient development and incorporation of software by/from other software vendors.

Note—If you are a vendor or contractor developing software for incorporation into standard frontpath distributions, you must follow these standards. If frontpath will be manufacturing units for you and you wish your software to be installed by frontpath, you must follow these standards.

3.1.1. Base Paths

The general Unix/Linux application development base path is: **/usr/local**. This directory normally contains, at a minimum, the subdirectories: **bin etc lib man sbin src**. In order to integrate ProGear development results with the development results of other value adding vendors, ProGear will use the "webzine" directory in **/usr/local**, resulting in a base path of: **/usr/local/webzine**. For your development, we recommend that you develop within: **/usr/local/<your_company_name>**. Other vendors will follow in like fashion.

3.1.2. Executables

Place all application type executables in: **/usr/local/<your_company_name>/bin**. This is acceptable since we use explicit pathing for execution. Place all system and kernel related executables in:

/usr/local/sbin	if they will run after system initialization completes
/sbin	if they are required for system initialization

3.1.3. Libraries

Static libraries may reside anywhere within the application source tree (e.g., **/usr/local/<your_company_name>/src**). Static libraries, such as those suffixed by ".a," are permanently linked with your program at Make time. Dynamic libraries, per Linux dynamic library management, must reside in:

/usr/local/lib	if they will be used by programs that will be run only after system initialization completes
/lib	if they will be used by programs that run at system initialization time

3.1.4. Configuration Files and System Control Scripts

The Unix/Linux standard location for system configuration data is: **/etc**. The Linux standard location for system initialization and termination scripts, hereafter referred to as system control scripts is: **/etc/rc.d**.

ProGear is using the 2.4.1 kernel and a Slackware 7.0 distribution. The Slackware distribution uses BSD Unix initialization **/etc/rc.d** for all system control scripts. Application configuration files are to be placed in **/usr/local/<your_company_name>/etc/**. Since Unix/Linux views application scripts as equivalent to executable programs, application scripts are to be placed in: **/usr/local/<your_company_name>/bin.**

Developers must adhere to the BSD format only. When faced with a package using SYS V initialization such as an RPM, locate the existing BSD location for your new package script changes, using the command: **grep _new_program_or_command_name_ /etc/rc.d/***, then, either:

1) Change the existing **/etc/rc.d/rc.xxx** script found with the above grep, or
2) Copy the **SYSV /etc/rc.d/init.d/yyyy** script to **/etc/rc.d/rc.yyyy** and edit as the new **rc.yyyy**, as needed for proper operation in the BSD initialization scheme.

3.1.5. Application Resources

You should place data used by your application or any media provided for users under **/usr/local/<your_company_name>/data.** If there is insufficient space on the partition containing **/usr/local**, then you may place your application resources on the data partition in **/data/<your_company_name>/data**. If your data is on **/data**, you should create a symbolic link to your installation directory, e.g.,
ln –s /data/<your_company_name>/data/usr/local/<your_company_name>/data

3.1.6. Patching a ProGear Distribution

The following is an example of files and scripts required to update the ProGear BIOS using the Remote Update utility (refer to Section 4.1 - Update Components for additional information on updating the BIOS).

Tar Ball File List:

CRUiSe/CRUiSeUpdate.sh – Update script called by the Client Component. This file must exist.
CRUiSe/ updateBIOS - BIOSUpdate script called by the update script. This could be included within the CRUiSeUpdate.sh script
CRUiSe/ pb_10218.rom - BIOS ROM File
CRUiSe/ CRUiSeManifest.txt - Manifest file created using md5sum and used by CRUiSe to verify the contents of the tar ball after download.

Script Contents:

CRUiSeUpdate.sh: This file is called by the Client Component during an update. In this case all it does is call the included updateBIOS script conditionally upon existence of the script in the expected location. It can be as complicated as necessary.

```
#! /bin/sh
#
# Start of CRUiSeUpdate.sh
#

if [ -e /tmp/CRUiSe/updateBIOS ]; then
. /tmp/CRUiSe/updateBIOS
fi

#
# End of CRUiSeUpdate.sh
#
```

updateBIOS:

```
#! /bin/sh
#
# Start of updateBIOS script
#

if [ -e /tmp/CRUiSe/pb_10218.rom ] && [ -x /usr/local/sbin/fpFlashBIOS ]; then
              /usr/local/sbin/fpFlashBIOS -w:pb_10218.rom
              shutdown -r now
fi

#
# End of updateBIOS script
#
```

Note—The tar ball contents must contain the "CRUiSe" directory in the path.

3.1.7. Source Code Packages (Including Device Driver Source)

Place these in: **/usr/local/<your_company_name>/src/<name_of_your_source_package>.**

3.1.8. Makefiles and Building Your Source

ProGear source code is assembled and compiled not only by the developer and package maintainer, but also by release engineering and potentially by other companies doing value added development for the platform. As a consequence, Makefiles must be as flexible as possible since the path environments that they will run in can be vastly different. Thus all Makefiles must construct their libraries and/or executables using relative paths as follows:

./	means this directory
../	means the parent directory of this directory

From within a source tree, higher level directories can be referenced by concatenating the "../" form as follows:
../../../ means three directories above this one

Exceptions: The only exceptions to the above are device drivers and system utilities and packages that ProGear development modifies such as X-Windows, whose pathing is a Unix/Linux standard and is hardcoded in the X build.

Each Makefile should support at least, the following targets:

Clean:	To clean out the build directory of all object and executable files. Syntax: `make clean`
All:	To build entire package without installing it. Syntax: `make` (in this case the target "all" is 'understood' by the make utility)
Install:	To install the results of a build Syntax: `make install`

Test the resulting executable(s) on or against the most current DevGear and ProGear distributions available before submission.

3.1.9. Source Code Package Documentation

All packages must contain at least one README that describes:

1) Package name and description
2) Where the output executes or is loaded from
3) How to build all executables, modules, scripts, and libraries. (This is the
4) "make" and "make demo" step.)
5) How to install the output from step 3. (This is the "make install" and "make install_demo" step.)
6) All required devices must be listed as follows:
    a) name
    b) type
    c) permission
    d) owner:group
    e) major number
    f) minor number or minor number range as needed
6) Any special owner, group or permission constraints that might prevent proper execution.
7) Any Linux or build component revision that this product is dependent on.
8) The development and demo/production distribution revision id & date this is intended for.

3.2. Running Local Web Servers on a ProGear

ProGear's Development software distribution provides a full Linux environment (requires a hard disk configuration) and as such, can host a web server, database server, scripting languages, and compiled code. The Development distribution comes with an Apache web server and the Perl scripting language pre-installed:

Apache is located in: **/var/lib/apache**
Perl is located in: **/usr/bin/perl**

As shipped, system initialization does NOT start the Apache daemon. You can start it locally with the command: **/var/lib/apache/sbin/apachectl start**, but be sure to review Apache's configuration files before doing so. You can set the system to start Apache automatically, at boot time, by including a reference to **/etc/rc.d/rc.httpd** in either **/etc/rc.M** or **/etc/rc.local**. Again, be sure that the Apache configuration is correct before doing so.

Rather than relying on the versions shipped with Slackware, you may wish to download and install the latest versions of Apache and Perl, from www.apache.org and www.perl.com, respectively. You can also extend Apache (e.g., by adding modules), load alternate web servers (e.g., Zope), load additional scripting languages (e.g., Python), install a database engine (e.g., MySQL), and install compiled code to be run as CGI scripts.

Note—To insure proper operation, we recommend compiling all such code directly on the ProGear.
Once you have the environment configured to your liking, you can load documents, scripts, compiled code modules (compiled on the ProGear), and data into the appropriate locations. Once loaded, you should be able to access the local web server with a URL of the form "**http://localhost/<document_path>**."

3.3. Developing Linux Applications

ProGear's Development distribution supports a full Linux environment, including C programming tools, and Perl as a scripting language. And of course, you can load additional tools, as you desire.

Programming for the ProGear is the same as programming for any Linux environment, except for the details of ProGear's input-output devices and associated drivers, which are detailed in Sections 5 and 6. The directory **/usr/local/webzine/src** contains examples of programs and Makefiles tailored to the ProGear environment.

3.4. Writing Browser Plug-Ins for the ProGear Environment

Netscape's plug-in architecture allows developers to extend the capabilities of Netscape Navigator. Plug-ins are software programs that extend the capabilities of Netscape Navigator in a specific way—giving you, for example, the ability to play audio samples or view video.

About Plug-Ins and MIME Types

Plug-ins are applications that run within the Netscape Navigator program to help interpret various file extensions and determine how to handle the files. Although they behave as though they are part of

Navigator, plug-ins are separate code modules that extend Navigator's capabilities by allowing it to support additional MIME types, play audio and video, display animations, and other features.

MIME (Multipurpose Internet Mail Extensions) media types specify most data types on the Internet, and consist of a major type (such as application, image, or text) followed by a minor type. If you create a new MIME type, you should register it with the Internet Engineering Task Force (IETF). Until the MIME type is registered, you should prefix it with x-. (For more information on MIME types, see RFC 2045 at: http://www.nacs.uci.edu/indiv/ehood/MIME/2045/rfc2045.html).

Once you have decided what to do with your plug-in, Netscape developers resources provide avenues to help make creating it a simple process. The following information is an excerpt from the resource webpage located at: http://developer.netscape.com/docs/manuals/communicator/plugin/index.htm.

1) Decide on a MIME type and file extension for the plug-in. See "Registering Plug-ins."
2) Download and decompress the SDK. See "The Plug-in Software Development Kit."
3) Create the plug-in project. You can either start from the template provided for your operating system in the source directory, or construct a plug-in project in your development environment using SDK-provided files. You can use a variety of environments to create plug-ins.
4) Make the necessary changes and additions to the SDK files to provide plug-in functionality and implement plug-in API methods for basic plug-in operation. You will find an overview of using Plug-in API methods at this location. More information about each function is also provided.
5) Build the plug-in for Linux / ProGear. See "Building Plug-ins."
6) Install the plug-in in the ProGear plug-in directory. See "Installing Plug-ins."
7) Test your plug-in and debug as necessary.
8) Create an HTML page and embed the plug-in object. For information about the HTML tags to use, see "Using HTML to Display Plug-ins." To see your plug-in working, simply display the HTML page that calls it in Communicator.

Upon completion Netscape also provides the ability to post and publish your plug-in. Refer to the URL below for listing your plug-in on Netscape's reference page:
http://home.netscape.com/plugins/plg_profile.html

3.5. Interfacing New Devices and Writing New Drivers

The following information provides a framework for developing a device driver for ProGear:

1. Develop and compile the new device driver as a kernel module for the current kernel used on ProGear. (As of this writing the kernel version is v2.4.1-ac19.)
2. Load the device driver onto ProGear. The recommended location for the device driver is: **/lib/modules/2.4.1-ac19/misc**
3. Run **depmode** to update the kernel dependency map in **/lib/modules/2.4.1-ac19/modules.dep**
4. At Linux initialization time, load the driver module into the ProGear Linux system with the **modprobe** command.

Note—It is beyond the scope of this document to discuss the technical development of Linux device drivers. There are many good sources on writing device drivers for Linux. See the Bibliography in Appendix D for references.

4. Remote Software Update

ProGear provides a way to automatically or manually update the system software through the Internet when new features become available. The software update is comprised of a three component system. The first of the components is server-based. It contains the policies for activating an update and ensuring its success. The second component resides on the client. It is a small executable that will retrieve the update and invoke the update process. The third component is the update package, a compressed file containing an update script and the files necessary for the update. Refer to Section 6.5.8 to make changes in the registry file for remote or manual software updates.

4.1. Components

Server Component
The server retains information about ProGear such as an MD5 checksum for the disk (or other equivalent non-volatile memory) contents, the MAC address of the unit, and the state of each unit. The update policies and control of the ProGear's state are also controlled by the server component. This component also uses information from the client (e.g., battery state) to determine when the update should occur.

Clients can communicate with the server using HTTP and the server component will be implemented in any appropriate language. It will also include a simple web-based UI for controlling the state and other update information (e.g., corresponding update file to download).

Client Component
This component resides on the ProGear. It is turned on either in the registry or the Pad Login Application, and is invoked at system initialization. The client requires an AC power connection. It will query the server for the update status and provide the MAC address, hard drive space and current version to the server in the request. Upon instructions from the server, this component will scan the hard disk (or other equivalent non-volatile memory) and return MD5 checksums to the server for files that are different than what was expected or missing from the file list. It will also take care of downloading the update file, decompressing it, verifying its contents using MD5 checksums, and then invoke the update shell script.

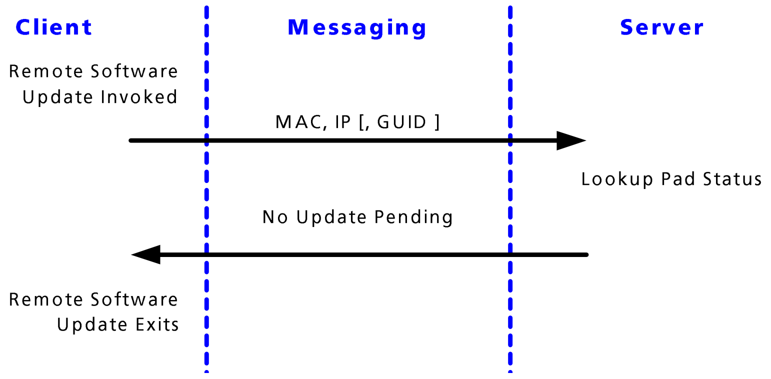
The client component will wait for the update script to complete. Upon completion of the update script, the client component will return the output of the update script to the server unless the update script initiates a reboot of the unit.

Update Component
A new update component will be created for each update. This component will have at least two files: the update script and the manifest. It may also contain any number of additional files required for the update. The update script will always be named CRUiSe_update.sh. The manifest will always be named CRUiSe_manifest.txt. Please see Appendix C for the format of the manifest file. The update component will packaged in a tar'red and compressed file with the update script and the manifest at the root level.

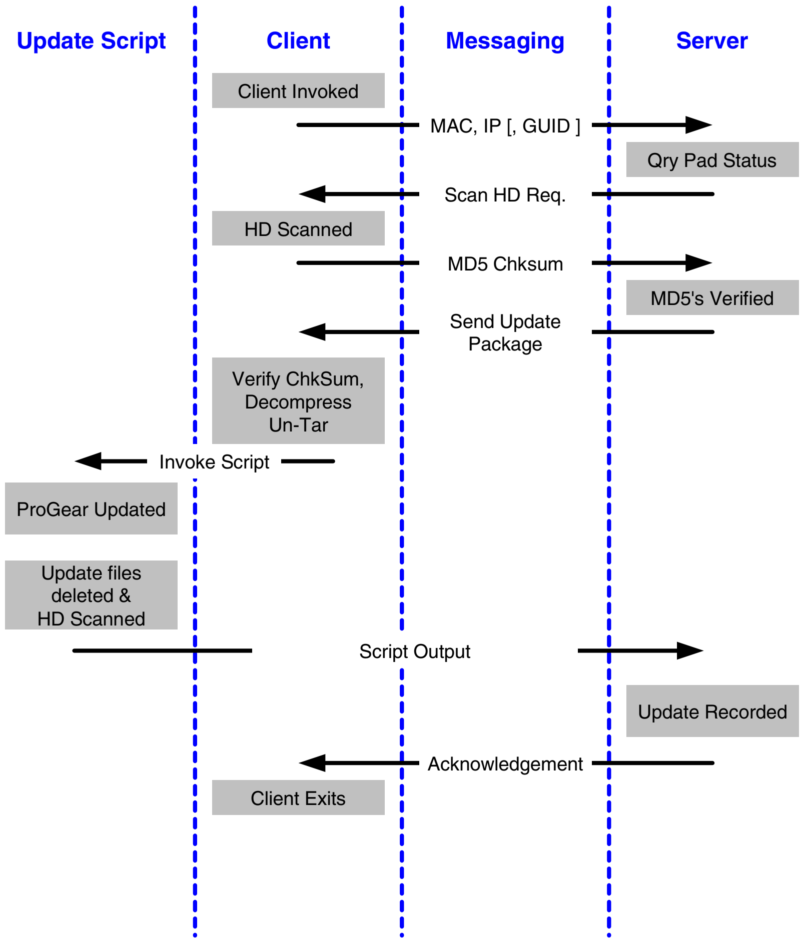
The update script will take care of updating the ProGear. It will disable the sleep mode and power button to prevent accidental interruption by the user. The update script will also return an error message in the event that a failure occurs, and clean up any files that it places on the system.

4.2. Sequence Diagrams

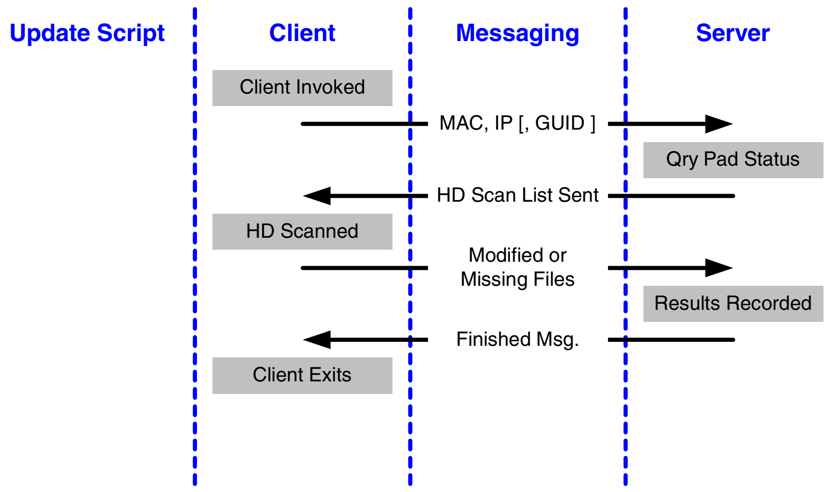
Sequence #1 – Example of No Update Needed



Sequence #2 – Example of a Successful Update



Sequence #3 – Example Disk Scan Request



4.3. Remote Software Update Specifications

File Formats

The remote software update has been setup to ignore the carriage-return at the end of lines in files created on MS-DOS/Windows machines. To avoid any problems, please create files for the remote software update on UNIX/Linux-based machines so they will have only a line-feed character to mark the end of a line.

HTTP Protocol

The remote software update communicates with the server using HTTP as its transport protocol and places its information such as the MAC address, file space, and messages, in the body of the message (an HTTP/1.0 POST message). Those data elements for the remote software update are sent in “messages” of the form X-FrontPath-[MSG]: [value] similar to the messages of a regular HTTP/1.0 message.

Since the remote software update specific messages are sent in the body of the HTTP message, most HTTP servers will ignore the messages and just pass them along to the script. Therefore, you can use almost any web server for the server side of the remote software update, but there is no need to know the actual messaging in order to make use of the remote software update.

Also, the server side of the remote software update is presently a CGI script running on an Apache HTTP server, and since HTTP is used over port 80, the remote software update will be able to operate over a firewall that is set up to allow normal web traffic. The remote software update can also be configured to use other TCP/IP ports if desired.

Remote Software Update Defaults

In the absence of a local configuration file (/etc/cruise.conf), the remote software update is hard-coded to contact http://cruise.frontpath.com on port 80. This file should NOT be included in the release image unless the default server needs to overridden. However, a better solution would be to allow a unit to contact the server and be recorded and set with the appropriate configuration settings. Also, when this file is absent, the remote software update will include that information in its messages to the server.

Identifying A Unit

The remote software update initially uses the MAC address as the means of identifying a unit. The MAC address is passed as a string of the form XX:XX:XX:XX:XX:XX where X is a hexadecimal digit. If desired, the remote software update server can generate a unique ID (similar to an MS Windows GUID) for each unit and then use that value as the primary means for recognizing a unit. The benefit is that once the unit receives this ID, the unit can still be uniquely identified by the ID. However, that ID is stored in the "**/etc/cruise.conf**" file, so if that file is ever removed or corrupted, the remote software update server will lose track of that units history. If that happens, it will just create a new ID and start a new history for the unit so that the ID is slightly better. Therefore, deletion of the file and file corruption are much less likely than exchanging network cards.

Remote Software Update Messaging

The remote software update contacts the server with a "mode" message that is considered a request. The server then replies with a "mode" directive indicating what action should be taken. For example, the client might make a request to do a hard drive scan, but the server, a newer version number, may instruct the client not to do anything by sending it a directive of CRUISE_FINISHED.

Remote Software Update Hard Disk Scans

When desired, the remote software update can be instructed to perform a scan of the file system and compare the MD5 checksums of the files on the system. To do this, the server will need a list of files and MD5 checksums to send down to the client. The client will then simply check the files on the list and report back any files whose checksums do not match or are missing from the system.

The list of files can be created quite easily using the md5sum utility on most Linux distributions. The remote software update itself does not need this utility because the MD5 algorithm is compiled directly into the remote software update. The files to check need to be specified with a full path so every file must begin with "/." If this is missing, the remote software update may either fail to find the files or find files other than the files intended. An example of a scan file looks like the following:

```
782ca80893ab4231782ca80893ab4231  /vmlinuz
782ca80893ab4231782ca80893ab4231  /etc/hosts
782ca80893ab4231782ca80893ab4231  /webzine/.netscape/preferences.js
```

Note—A regular expression of the form "^([0-9a-z]{32})[ ]+(.*)$" is used to capture the information from each line. The following is an example of how to create one of these files using the "find" and "md5sum" utilities: "`find /etc -exec md5sum {} \;`"

Remote Software Update Files

An remote software update file is nothing more than a "gzip" compressed TAR file (.tgz or .tar.gz) with two special files inside of it (an update script named CRUiSeUpdate.sh and CRUiSeManifest.txt) and a directory structure with "CRUiSe" as the parent directory. The update file will be decompressed into /tmp, so the update script and the manifest file are expected to be found in "**/tmp/CRUiSe/CRUiSeUpdate.sh**" and "**/tmp/CRUiSe/CRUiSeManifest.txt**" respectively. You can include as many other files as needed for the update (memory permitting) in the main update file.

The remote software update program will check only the manifest to compare the MD5 checksum of each of the files, and will ignore them otherwise. Only the update script needs to know about the files and what should be done with them. If the MD5 checksums do not match, the remote software update will report an error back to the server and will not run the script. But if all the checksums are OK, then the remote software update will run the update script, capture its output, and return the output to the server for the server to decide if the update occurred correctly. An update must be accompanied by a definitive explanation of how one determines the success or failure of an update. The results can also be emailed to any interested parties if so desired.

Notes—A minimal update tar file would contain the following (note the absence of a leading "/" so the paths are relative):

```
CRUiSe/CRUiSeUpdate.sh
CRUiSe/CRUiSeManifest.txt
```

At this time, the remote software update will read the update file from a socket (1024 bytes at a time) and pipe it directly into the "tar" program using the following command "`tar -xvz -C /tmp`."

The update script must take care of cleaning up after itself. The remote software update program will not remove any files after the update completes. The update file may contain sub-directories if desired. Since the checksum of a file cannot be predicted, the CRUiSeManifest.txt file must not appear as an entry in the manifest file itself. The server will create and transmit an MD5 checksum for the update file when it sends one. The remote software update will then check that MD5 checksum to ensure the compressed tar file arrives intact.

4.4. Testing the Remote Software Update

There are several tools available for testing the remote software update. The following sections outline how to set up a test environment for the remote software update program.

Remote Software Update Server

- Set up an Apache server on Linux and create a "/cgi-bin" directory if one does not already exist.
- Install the "cruise.pl" Perl script in the "/cgi-bin" directory.
- To test the remote software update, tweak "cruise.pl" for your desired test scenario.
- To test update, point "cruise.pl" to where to find the update package.
- To test hard drive scans, you will need to point "cruise.pl" to where to find the hard drive scan file with the list of files and MD5 checksums.
- Next, open a window on the server (xterm or telnet window) and monitor the server's error log file (e.g., "tail –f /var/log/httpd/error_log").

Remote Software Update Client

- Install either the debug or remote software update release in the desired directory. The remote software update program can be run from any directory.
- Edit the configuration settings in the registry to point the remote software update program to the test server. An example of these settings is listed below.

Entries in the Registry:

```
# 0 is auto run remote update DISABLED
# 1 is auto run remote update ENABLED
AutoRemoteUpdate=0
# Remote Update Server
RemoteUpdateServer=cruise.frontpath.com
# 0 is Manual Update button in Toolbox DISABLED
# 1 is Manual Update button in Toolbox ENABLED
EnableToolboxRU=0
# 0 is NO Manual Remote Update on next reboot
# 1 is Manual Remote Update requested on next reboot
ManualUpdateReq=0
```

Remote Software Update Scripts

- To create an update script, first create a directory named CRUiSe.
- Create whatever subdirectories you need within CRUiSe, and add the files you need for the update as appropriate.
- Create a script file inside the CRUiSe directory and name it CRUiSeUpdate.sh. Be sure this script begins with the appropriate "magic cookie" line (e.g., #!/bin/bash).
- You will then need a "CRUiSeManifest.txt" file, which is a list of all the files in the update package. Do not attempt to include an entry for the CRUiSeManifest.txt file because you will not be able to predict the MD5 checksum for this file.
- Next, create a compressed "tar" file from the parent directory of the "CRUiSe" directory.

Note—A script file named "mkupdate" will create the manifest file, the compressed tar file and install the files in the directory "/home/httpd/cgi-bin/update.tgz." Simply comment out any lines, such as the last line, if you would rather perform that step manually.

Performing a Test of the Remote Software Update

Now that you have the server properly configured, the client configured, and the update script properly installed where the remote software update server can find it, run the remote software update (e.g., ./cruise).

Check the server log to see what happened on the server side and then check the client to make sure the client system was properly updated. Debug and repeat as necessary.

4.5. HTTP Messaging

The remote software update uses HTTP messaging to communicate between the client and server. See Section 4.3 for information on the HTTP protocol.

Note—The remote software update uses the GNU C library regular expression routines. See man 7 regex for details on the regular expression syntax.

Remote Software Update Modes

The Cruise modes are defined as an enumerated type set as follows:

```
CRUISE_DO_NOTHING_MODE,          //     0
CRUISE_FINISHED,                 //     1
CRUISE_CHECK_WITH_SERVER,        //     2
CRUISE_UPDATE_MODE,              //     3
CRUISE_HDSCAN_MODE,              //     4
CRUISE_ERROR_OCCURRED,           //     5
CRUISE_GUID_GEN_REQUEST,         //     6
CRUISE_UPDATE_CONF_FILE          //     7
```

The Remote Software Update Sends A Message to the Server

When the remote software update is first executed it checks for configuration file settings in the registry, which consists of the remote update server and whether the cruise process should be initiated at boot time.

The client will only send CRUISE_CHECK_WITH_SERVER during a single session. The client will read the remote update server address from the registry and initiate a cruise update.

Here is an example cruise configuration file:

```
    CruiseHostName: "cruise.frontpath.com"
    CruisePort: "80"
    CruisePadID: "000039840318-0310B45B-A4CEDF4957C8D777FB5CCBDF1CA51003"
    <blank line>
```

Note—The file must end with a blank line in order for the last text line to be read correctly.

The remote software update will then build a general HTTP header for all the messages it will send to the server. The general header will look like the following example:

```
    POST /cgi-bin/cruise.pl HTTP/1.0
    Host: <server host name>
```

The remote software update will then check for a Software Version Information File in /home/webzine/dist_version. If the file exists, then it uses the regular expression ”ˆVersion -[ ]+(.*)[ ]*$” to find the version information line: e.g., ”Version - 2.03.26-P.” Parsing stops as soon as a match is found for the regular expression. Lines not matching the regular expression are ignored.

Here is an example of the first few lines of a SW Version Info file:

```
    ProGear(tm) - Software Distribution
    Version - 3.00.20-P
    [snip]
```

The remote software update then looks for the MAC address of the NIC (Network Interface Card) by parsing the output of the ”ifconfig” program. The remote software update uses the regular expression ”.*ethernet[ ]+ `FrontPath-PadID: 000039840318-0310B45B-A4CEDF4957C8D777FB5CCBDF1CA51003`
```
    X-FrontPath-MAC: 00:00:39:84:03:18
    X-FrontPath-TimeStamp: Wed Feb 28 01:12:14 2001
    X-FrontPath-PadSWVersion: 2.03.26-P
    X-FrontPath-ClientMode: 2
    X-FrontPath-RAM: 124,3
```
`X-FrontPath-SWAP: 549,544`hwaddr[ ]+(([0-9a-f]{2}:){5}[0-9a-f]{2})” to locate the appropriate line.

Here is an example of the line from ifconfig that the remote software update is expecting:

```
    wlan0      Link encap:Ethernet  HWaddr 00:50:BA:77:1F:B2
```

The remote software update will also gather the following system information to send to the server: total RAM, total free RAM, and total free disk space. The values are in Mega-Bytes.

Next, the remote software update will assemble this information and send it to the server in a message that will look like the following example:

```
    POST /cgi-bin/cruise.pl HTTP/1.0
    Host: cruise.frontpath.com
    Content-Length: 194
    X-FrontPath-MAC: 00:00:39:84:03:18
    X-FrontPath-TimeStamp: Wed Feb 28 01:14:07 2001
    X-FrontPath-PadSWVersion: 3.00.20-P
    X-FrontPath-ClientMode: 6
```

```
X-FrontPath-RAM: 124,3
X-FrontPath-HD: 549,544
```

The PadID will also be sent to the server if a properly formatted cruise.conf file exists and it contains a CruisePadID line. The message will look like the following example:

```
POST /cgi-bin/cruise.pl HTTP/1.0
Host: cruise.frontpath.com
Content-Length: 268

X-
```

The Remote Software Update Server Replies to the Client

The remote software update server will look at the mode sent by the client and may use that information or discard it at its option. The client will, however, respect the request sent by the server.

If the server does not want the client to do anything, it will send a CRUISE_FINISHED or a CRUISE_DO_NOTHING_MODE as the mode in the message. It would be proper, but is not required, to use CRUISE_DO_NOTHING_MODE when the server has nothing for the client to do, and use CRUISE_FINISHED after the server has requested that the client perform a task. Either one will cause the client to exit. The message received will look like the following example:

```
HTTP/1.1 200 OK
Date: Wed, 28 Feb 2001 02:30:12 GMT
Server: Apache/1.3.12 (Unix)  (Red Hat/Linux) mod_perl/1.21
X-FrontPath-ContentLength: 0
X-FrontPath-MD5CheckSum:
X-FrontPath-UpdateMode: 0
Connection: close
Content-Type: text/html
```

If the server wants the client to perform an update, the mode will be set to CRUISE_UPDATE_MODE. The MD5 check-sum and the content length will be set appropriately for the data portion of the message and the update (a compressed tar file) will follow the last line of the HTTP. The message received will look like the following example:

```
HTTP/1.1 200 OK
Date: Wed, 28 Feb 2001 02:30:12 GMT
Server: Apache/1.3.12 (Unix)  (Red Hat/Linux) mod_perl/1.21
X-FrontPath-ContentLength: 1234
X-FrontPath-MD5CheckSum: 12345678901234567890123456789012
X-FrontPath-UpdateMode: 3
Connection: close
Content-Type: text/html

< contents of compressed tar file in binary form >
```

If the server wants the client to do a hard disk scan, the mode would be set to CRUISE_HDSCAN_MODE. The MD5 check-sum and the content length will be set appropriately for the data portion of the message, and the list of files to check along with their MD5 check-sums will follow the last line of the HTTP headers. The message received will look like the following example:

```
HTTP/1.1 200 OK
Date: Wed, 28 Feb 2001 02:30:12 GMT
Server: Apache/1.3.12 (Unix)  (Red Hat/Linux) mod_perl/1.21
X-FrontPath-ContentLength: 1234
X-FrontPath-MD5CheckSum: 12345678901234567890123456789012
X-FrontPath-UpdateMode: 4
Connection: close
```

```
Content-Type: text/html

358c626ffa49597a9e28bd8987c80bf8  /usr/bin/infocmp
2202dcfbd7fde4239a0d38c091818800  /usr/bin/infotocap
90c75c3c4d82174ddaa8ac11c48c012e  /usr/bin/reset
37b879cb115edd118208af55b47bec5a  /usr/bin/resetall
cac14c100ebb8a5510ce3e16408386ec  /usr/bin/tack
2202dcfbd7fde4239a0d38c091818800  /usr/bin/tic
547375fce5222e956fe6cbed0ac59bcc  /usr/bin/toe
```

After the hard disk scan list is sent from the server to the client, the client will perform a scan of the hard disk. The client will then display a list of missing files or any files whose checksum did not match. A blank list displayed by the client indicates that all items requested to be checked by the server are present and OK.

The client will then carry out the request from the server and reply with either an error if one was detected or it will send back the output of the CRUiSeUpdate.sh shell script, unless the update script kicks off a restart. In this case there will be no reply from the client. If an error occurs, the client will send back messages similar to the following:

```
X-FrontPath-ErrorOccurred: 1
X-FrontPath-Error-Msg: Could Not Open Config File For Writing!
```

The default location for the remote software update server is http://cruise.frontpath.com:80, and the default location of the Cruise server program is **"/cgi-bin/cruise.pl**." The server address and port can be overridden in the cruise.conf file, but the program location can not be overridden at this time.

5. ProGear Hardware Reference

5.1. Hardware Specification

CPU: Transmeta TM3200 x86-compatible processor with integrated Northbridge	**Graphics Processor:** Silicon Motion LynxEM4 with 4MB embedded memory, supports accelerated BLT's, line, and hardware rotation	**ALI 1535 South Bridge:** supports USB, PCI IDE controller, SM Bus interface for battery control, real-time clock, AC97 2.1 audio support
Memory: • 1 MB flash memory boot ROM • 64 MB SDRAM or 128 MB SDRAM (16MB memory used by Transmeta's CMS (Code Morphing Software)	**Storage:** • 64 MB compact flash or • 5-6 GB hard disk drive (hard drive size is approximate and may vary during manufacturing builds)	**Display:** • 10.4" Toshiba Low Temp Poly Silicon TFT LCD • 1024 x 768 XGA (standard) • 800 x 600 SVGA (optional) • 18-bit resolution, 65k colors • Software controlled backlight
Size: 9" x 11" x 1" **Weight:** 3.2 lbs. **Touch Screen:** Fault tolerant 1k x 1k points, 5-wire resistive **Scroll button pointing device** **AC/DC Power Supply 8-15v. 3.3amp** **Power Switch:** software controlled system state change switch on the end of the unit **Reset Switch:** backside of ProGear (activates a hardware reset)	**Audio Output:** • AC/97 compliant Codec • Software controlled volume and mute • Monaural speaker • Stereo output: 1/8" stereo jack, supports headphones or amplified speakers • Internal microphone. Microphone jack, 1/8", tip and ring bias, supports 600 ohm dynamic and 10k ohm electric condenser microphones	**PC Card Type 2 Slot:** PCI 1410 single slot PC Card CardBus controller, compliant with '97 PC Card standard. Supports hot card insertion and removal. Normally intended to have an 802.11b wireless card installed. ProGear units are available in both an open slot configuration for client customization, and an "integrated" card version where the PC card slot is covered and used for the integrated 802.11b solution.
Ports: one USB port, one IrDA v1.1 (MIR & FIR) high-speed infrared port (Developers version also has PS/2 mouse and keyboard ports and a VGA video port).	**Battery:** rechargeable SMART Lithium-Ion, 1600 mAh 3-cell battery, upgradeable to 6-cell battery **Swap Battery (AAA battery):** allows main ProGear battery to be swapped while unit is off without losing memory	**Environmental:** • Operating temperature: 0° C(32°F) to 35°C (94°F) • Altitude up to 10,000 feet • Humidity to 95% at 30 degrees C (85°F) • Regulatory: FCC-B, UL, CE Mark

5.2. Cradle Specification

- Holds screen at a 30 degree (approximate) viewing angle
- Supports only landscape orientation
- Hot-swappable system connection between ProGear and Cradle
- Adds four USB ports
- Provides power to ProGear
- Charges ProGear's internal battery
- Charges an extra ProGear battery
- Displays Cradle Status
- Green LED: Cradle connected to AC Power
- Red LED: An extra battery is recharging
- Power Requirement: 15V, 3.3A (uses the ProGear Power Adapter)

5.3. Cradle LC Specification

- Holds screen at a 30 degree (approximate) viewing angle
- Supports only landscape orientation
- Hot-swappable system connection between ProGear and Cradle
- Adds one USB port
- Provides power to ProGear
- Charges ProGear's internal battery
- Displays Cradle Status
- Left Green LED: Cradle connected to AC Power
- Right Green LED: ProGear is connected to Cradle LC
- Power Requirement: 15V, 3.3A (uses the ProGear Power Adapter)

6. ProGear Software Reference

6.1. Overview

The baseline ProGear software environment consists of:

- The Phoenix BIOS (in boot PROM)
- The Slackware 7.1 Linux distribution
- A mobile Linux kernel, version 2.4.1-ac19 (located in **/boot/bzImage-2.41ac19**)

ProGear supports three standard distributions:

- ”DevGear” development distribution contains a full Linux distribution, as well as the gnu development tools and source code for much of the ProGear software.
- ”ProGear” distribution is a stripped down set of files intended only to support wireless Web access.
- ”ProGear-TC” (thin client) distribution is small enough to be loaded into compact flash memory in a diskless configuration. This version is essentially the same as ProGear except that some files are removed.

The following sections discuss how ProGear’s distributions differ from the standard Slackware distribution. Additional details can be found in the commented source code and other documents found in **/usr/local/webzine/src** of the Development distribution, which contains ProGear specific source, build, configuration, and documentation files.

Note—Section 3.6 discusses how to tailor a distribution to your needs.

6.2. Flash Memory and Disk Partitioning

6.2.1. Development Distribution

Hard Drive Partition Architecture

Running the command “`fdisk -l /dev/hda`” will produce the following:

Device	Boot	Start	End	Blocks	Id	System	Mount Point
/dev/hdb1	*	1	17	136521	83	Linux native	/
/dev/hdb2		18	34	136552+	82	Linux swap	-
/dev/hdb3		35	47	104422+	83	Linux native	/tmp
/dev/hdb4		48	608	4506232+	5	Extended	-
/dev/hdb5		48	60	104391	83	Linux native	/var
/dev/hdb6		61	226	1333363+	83	Linux native	/usr
/dev/hdb7		227	367	1132551	83	Linux native	/usr/local
/dev/hdb8		368	608	1935801	83	Linux native	/tooling

Directory Structure

Root Directory Tree

/bin	/boot	/cdrom	/dev	/etc	/home
/lib	/lost+found	/mnt	/opt/	/proc	/root
/sbin	/shlib	/tmp	/tooling	/usr	/var

6.2.2. ProGear Distribution

Hard Drive Partition Architecture

Running the command "`fdisk -l /dev/hda`" will produce the following:

Device	Boot	Start	End	Blocks	Id	System	Mount Point
/dev/hdb1	*	1	17	136521	83	Linux native	/
/dev/hdb2		18	34	136552+	82	Linux swap	-

Directory Structure

Root Directory Tree

/bin	/boot	/dev	/etc	/home	/lib	/lost+found
/mnt	/proc	/root	/sbin	/tmp	/usr	/var

6.2.2.1. Adding a Partition to the ProGear Distribution

If you require more space on the ProGear, you can add one or more new partition(s) to take advantage of the space not currently used.

To create the new partition(s) you will need to perform the following steps:

1. Run "/sbin/fdisk" to create the new partition(s).
2. Run "/sbin/fdisk –l /dev/hda" and write down the block size of new partition(s).
3. Run "/sbin/mke2fs /dev/hda[partition number] [block size]"
4. Create a directory as a mount point.
5. Add the new partition and mount point information to /etc/fstab.
6. Reboot.

6.2.3. ProGear Flash Drive Distribution

Hard Drive Partition Architecture (Future Product - Based on a Compact Flash size of 128MB)

Running the command "`fdisk -l /dev/hda`" will produce the following:

Device	Boot	Start	End	Blocks	Id	System	Mount Point
/dev/hdb1	*	1	9	72261	83	Linux native	/
/dev/hdb2		10	18	72292+	82	Linux swap	-

Directory Structure

Root Directory Tree

/bin	/boot	/dev	/etc	/home	/lib	/lost+found
/mnt	/proc	/root	/sbin	/tmp	/usr	/var

6.3. System Initialization, Off, and Shutdown Sequences

6.3.1. Initialization

On power-up, ProGear goes through standard BIOS and Linux initialization sequences, ending with execution of **/sbin/init**, which performs final initialization as directed by the contents of **/etc/inittab**.

In both the Development and ProGear distributions, the inittab directs invocation of **/etc/rc.d/rc.S** to initialize device drivers and other system-level functions. In the ProGear, this includes starting the touch screen driver, scroll button driver, and power button watch routine.

Finally, the inittab runs the initialization script for the default runlevel (specified in inittab):

- Development distribution: this is Slackware's default of runlevel 3 (multi-user, text interface), initialized by **/etc/rc.d/rc.M**, which leaves the system presenting a textual login prompt.
- ProGear distribution: the default is a ProGear-defined runlevel 5, initialized by the script in /usr/local/webzine/X/startOurX. The script su's to user "webzine," starts X-windows, starts a routine to read the scroll button, starts Netscape, and creates a Menubar. The full sequence is detailed in /usr/local/webzine/X/startOurX, and the various configuration files in /home/webzine.
- Netscape is started in **/usr/local/webzine/apps/dt_menubar/dt_menubar** using http://www.frontpath.com as default Home page if no page is specified in /home/webzine/.netscape/preferences.js.

Note—For additional details on the distributions, see /etc/inittab and the files in /etc/rc.d/.

6.3.2. Off and Shutdown

Off and Shutdown are activated by the unit's power button, or by software action, as follows:

- The catch_button program is started by root in the rc.S script.
- The catch_button program will react to three events: a single push of the power button, a double push (with one second separation) of the power button, and the existence of the file /tmp/.softS1.
- If a double push of the power button is detected, catch_button will execute the program shutdown_special, using an absolute pathname to /etc/rc.d/shutdown_special. The shutdown_special program will call the halt script, using an absolute pathname to /etc/rc.d/halt. The halt script will execute the power down program, using an absolute pathname to /etc/rc.d/powerdown. The power down program will put the system into S5 (soft OFF) power state.
- If a single push of the power button is detected, or the file /tmp/.softS1 appears, catch_button will call the doS1Sleep script. The doS1Sleep script will call the goIntoS1 program. The system will then go into the S1 (Power ON Suspend) power state. When the power button is pushed, the system will resume out of S1, back to S0 (ON). The goIntoS1 program will complete, the doS1Sleep script will complete, and control will return to the catch_button program, which will resume polling for the aforementioned three events.

  This single button push event will shut down the following unit functions:
  - LCD backlight
  - HDD
  - PC Card slot (RF)
  - Various clocks such as the CPU clock for maximum power savings
- The goIntoS1 program will check for a subsequent power button push just before going into S1. If found, the S1 transition is stopped, and an S5 transition is started via shutdown_special (see above).
- The catch_button program will also initialize the SMBus interface.
- Forced Shutdown occurs by pressing the power down button for four seconds.

6.4. Run-Time Environment

After initialization is complete, the run-time environment consists of a standard Linux environment, with the following modifications and extensions.

6.4.1. Display and Backlight

Display access is via standard Linux devices and X-windows functions. The LCD dimmer is normally controlled through the ProGear toolbar, using the code in /usr/local/webzine/apps/dt_toolbox/dt_toolbox. The scrollbar under the dt_toolbox is the only way to change the brightness at this time.

6.4.2. Touch Screen

The tscreenDriver kernel module, and usr/local/webzine/X/inputhandler3, both started during system initialization, read data from the touch screen and send data to the X server using the pipeinput.so module. Touch screen events are normally processed by the X server as mouse events (pen up/down and x/y coordinates). Detailed documentation and examples are available in /usr/local/webzine/src/inputhandler.

6.4.3. Scroll Button

Scroll button events are processed by the kernel module n9joyDriver, loaded at system initialization. The scroll button sends events via X-input as keyboard button events and behaves like a single button mouse. A detailed theory of operations can be found in /usr/local/webzine/src/n9joy/README.TOO.

6.4.4. Audio

Audio functions are processed by kernel modules gfxPM and trident, which are loaded at boot time.

6.4.5. Power Button

The power button is monitored by /etc/rc.d/catch_button, which is started during system initialization. Section 6.3.2 provides details.

6.4.6. PC Card Slot

PC Card slot functions are started during system initialization, using configuration data stored in:

```
/etc/pcmcia/config
/etc/pcmcia/config.opts
```

Network configuration is stored in:

```
/etc/pcmcia/network.opts
```

Additional configurations for the Lucent Orinocco Wireless LAN card are stored in:

```
/etc/pcmcia/wireless.opts
```

Additional configurations for the GemTek Wireless LAN card are stored in:

```
/etc/pcmcia/wlan-ng.conf
/etc/pcmcia/wlan-ng.opts
```

Note—ProGear's hardware provides the signal paths necessary for a PC Card to wake up the system from sleep mode. However, to extend battery life, ProGear's software currently removes power from the PC Card slot when the unit is sleeping. Thus, PC Cards cannot in practice, wake up the unit.

6.4.7. USB Port

The USB driver is a kernel module, loaded at boot time.

6.4.8. Real-Time Clock

Real-time clock functions are provided on the ALI 1535 South Bridge chip. To access the hardware clock on the ProGear use the Linux utility "hwclock." Hwclock is a tool for accessing the Hardware Clock. You can display the current time, set the Hardware Clock to a specified time, set the Hardware Clock to the System Time, and set the System Time from the Hardware Clock. Detailed information on hwclock can be found at:
http://www.linux.gr/cgi-bin/man2html/usr/share/man/man8/hwclock.8.gz.

6.5. X-Windows and ProGear User Interface

6.5.1. ProGear Modifications to X-Windows

ProGear's X-server is based on Xfree86, with modifications to support the touch screen and scroll button, and use the Silicon Motion graphics accelerator. Details can be found in /usr/local/webzine/src/xc/README.X11.

6.5.2. Invoking X-Windows

In the ProGear software distribution, X-windows is started during the last step of system initialization, as described in Section 6.3.1. X-windows is invoked as user "webzine," thus initialization is controlled by the .xinitrc file located in /home/webzine. In the development distribution, initialization completes by displaying a textual login prompt. To start X-windows, you can either:

1) log in as user "webzine" and then execute "./startx," which will initiate the normal user environment or,

2) login as root, then type "startx" to invoke X-windows, using the .xinitrc file located in /root

6.5.3. Window Manager

ProGear uses the Little Window Manager. Configuration files are found in /usr/local/webzine/apps/dt_lwm/dt_lwm.

6.5.4. Menubar

During normal operations, a Menubar is always displayed across the bottom of the screen. Other windows may not overlap the Menubar. The Menubar displays icons that provide access to:

- frontpath's icon launch for online support
- Toolbox (see Section 6.5.5)

- Virtual keyboard and handwriting recognition (see Section 6.5.6)
- Netscape
- Network connection status, mute status, battery status, time and calendar
- Close the active Netscape window (instance)
- Desktop Manager - controls which open window or application is active and on top
- Put ProGear into Standby mode

6.5.5. Toolbox

The Toolbox utility, launched from the Menubar, allows modification to system preferences, including:

- Setting screen brightness
- Advanced Toolbox settings
- Invoking a calibration utility to re-calibrate the touch screen
- Setting portrait or landscape display mode
- Adjusting the audio volume
- Adjusting the runtime power settings
- Selecting the handwriting position
- Manual Cruise update
- Activating diagnostics
- Muting audio output

6.5.5.1. Advanced Toolbox Settings

A password-protected utility, launched from the Toolbox, allows modification to advanced system preferences, including:

- Setting the RF wireless network/SSID name
- Enable/disable wireless encryption
- Enable Network Time Protocol
- Enable/disable Cruise software update
- Set Cruise software update server address or name

6.5.6. Text Input

The dt_input routine, invoked during system initialization supports text entry through either a virtual keyboard or handwriting recognition. Display of the virtual keyboard or handwriting recognition windows is invoked through the Menubar. Data flows from the touch screen, through the tscreenDriver kernel module, through the inputhandler3 routine, and into the X-server via the pipeinput.so module.

6.5.7. Support Files Functions

The following files provide support for various functions:

- /usr/local/webzine/etc/dt_registry_1024x768.data, dt_registry_1024x768.data.DEFAULT, dt_registry_800x600.data, dt_registry_800x600.data.DEFAULT - contains settings used by the Menubar, Toolbox, dt_advsetup, dt_clock, dt_battery, dt_diagnostic, cruise, and dt_input
- /usr/local/webzine/apps/dt_connect/dt_connect - displays network connection status, also wireless signal quality if the information is available
- /usr/local/webzine/apps/dt_init/dt_init - initializes registry data
- /usr/local/webzine/apps/dt_cleanup/dt_cleanup - runs upon X initialization and cleans out the cache and tmp directory to free up disk space
- /usr/local/webzine/apps/dt_advsetup/dt_advsetup - user interface for setting up network connection, clock, and other system configurations
- /usr/local/webzine/apps/dt_clock/dt_clock - clock and calendar utility
- /usr/local/webzine/apps/dt_battery/dt_battery - displays battery status and power configuration
- /usr/local/webzine/apps/dt_diagnostic /dt_diagnostic - displays system information, allows the you to test memory, audio, and touch screen sub-system
- /usr/local/bin/cruise - ProGear remote update utility

6.5.8. Registry Files

This section discusses the software registry on ProGear. The registry is used by the ProGear platform software, drivers, background tasks, and X-level applications. System settings and customized configurations are stored in the registry. Software uses the registry to save and retrieve data. It can also be used to share information between applications. The information that follows outlines the format of the registry and the predefined registry keys.

The ProGear registry contains four files: ”dt_registry.data.800x600.DEFAULT” and ”dt_registry.data.1024x768.DEFAULT” are master files that come with the distribution. The ”dt_registry.data.800x600” and ”dt_registry.data.1024x768” are working files that are generated the first time ProGear runs. The .DEFAULT file also serves as the backup file. In case the working file is corrupted, the system can be recovered from the .DEFAULT file. To modify the registry of a distribution, changes should be made to the .DEFAULT file.

When ProGear starts, registry data will be copied to the memory. Applications that use the registry will read and write from the memory only. Memory copy of the registry will be flushed to the file periodically and when ProGear is turned off. Changing the registry file while ProGear is running has no affect on the operation.

6.5.8.1. Registry File Format

The first line of the registry file contains the version number. It should never be modified.

# - comment
This must be the first character of the line. The Registry ignores the complete line that follows it.

* - section name
Defines the start of a section. The section is terminated by the definition of a different section or end of file.

key=value
The key and its associated value

In any case, space is not ignored and is considered part of the name or the value. Also, the registry is case sensitive. A key ”Progear” is different from a key ”ProGear.”

Here is an example:

```
...
#This is an example of the ProGear registry file format

*first section
first key=string value
integer key=1234

*next section
first key=no

frontpath=lower case
Frontpath=upper case
...
```

In this example, both the ”first section” and ”next section” have the same key named ”first key,” but they are not related. In ”next section”, both ”frontpath” and ”Frontpath” are unique and valid keys.

6.5.8.2. Predefined Registry Keys

The reserved registry key has a predefined value. The value is vital to the operation of the ProGear, and it should not be modified from its default setting. The user configurable registry keys allow the customer to modify the standard configuration.

Reserved Registry keys

1.1. section ProGear
    HomeDirectory
        - Directory for the default user
    AppsDirectory
        - Main directory where ProGear applications reside
    frontpath Link
        - An HTML page that Netscape will start with when you tap the frontpath icon on the Menubar

1.2. section Screen
    LCDwidth
        - Physical screen width in number of pixels
    LCDheight
        - Physical screen height in number of pixels
    Width
        - Display width in number of pixels. It changes when the orientation changes
    Height

- Display height in number of pixels. It changes when the orientation changes

Orientation
- Screen orientation. 0 is landscape, 1 is right portrait, 2 is left portrait

1.3. section Preference

Border
- Window frame size in pixels.

DefaultMenubarUp
- Initial state of Menubar when ProGear first powers up 0 is hidden, 1 is visible

DominantHand
- When ProGear is in landscape orientation, text input application can be set to either side of screen. 0 is left side, 1 is right side.

VolumeMute
- Turns off the audio output from speaker and headphone jack. 0 is NOT mute, 1 is mute

MasterVolume
- Master volume controls both built-in speaker and headphone jack. The range is in 0 – 100.

PowerSavingMode
- It sets the power saving mode. 0 is disabled, 1 is minimal, 2 is average, 3 is aggressive

Brightness
- It sets the intensity of LCD backlight. The range is in 0 –100

1.4. section MenuBar Data

Landscape Width
- Width of the Menubar in landscape orientation

Landscape Height
- Height of the Menubar in landscape orientation

Portrait Width
- Width of the Menubar in portrait orientation

Portrait Height
- Height of the Menubar in portrait orientation

1.5. section Utility Data

Landscape Width
- Width of the platform application (toolbox, keyboard) in landscape orientation

Landscape Height
- Height of the platform application (toolbox, keyboard) in landscape orientation

Portrait Width
- Width of the platform application (toolbox, keyboard) in portrait orientation

Portrait Height
- Height of the platform application (toolbox, keyboard) in portrait orientation

1.6. section Window Data

0=…
1=…
…

- Each key is a number, starting from 0, 1, 2, … The value of each key is the window class name of an application. A separate section, which is also the class name, has information that window manager uses to control the layout. If an application has a specific layout requirement, then an entry should be set up in section Window Data.

1.7. section ”window class name”

Attributes:

- 1 window is movable
- 2 window is resizable
- 4 window runs in the ”utility” area
- 8 window runs in the ”menubar” area
- 16 window should completely occupy client area
- 32 window attribute for touch screen calibration
- 64 window that can run in the background

AppName

Activated

WindowID

- This information is application specific

1.8. section Dt_battery

ThresholdWarn

- When the power level reported by smart battery is dropped below ”ThresholdWarn,” a message warns the user to recharge the battery. The range is 0% – 99%.

ThresholdShutdown

- When the power level reported by smart battery is dropped below ”ThresholdShutdown,” a warning message prompts the user to prepare for system shutdown. The range is 0% – 99%.

User Configurable Registry Keys

2.1. section Network

SSID

- The name of the Wireless Local Area Network

EncryptionKey

- WEP Encryption Key Values. Five characters for 64-bit key, 13 characters for 128-bit key

LoginID

- The ID is required for the user to log into Advance Setup

EnableEncryption

- Set the key to 0 to disable WEP encryption, 1 to enable WEP encryption

EnableNTP

- Set the key to 1 to automatically synchronize the ProGear time with the Network Time Protocol(NTP) Server, which is set in the key below. Set it to 0 to manually set the time.

NTPServer

- Host name or IP of the NTP Server

StatusSite

- Host name or IP of the Server that is setup to be the ping target. If the communication is dropped between ProGear and the server, the network status reports offline.

AutoRemoteUpdate

- Set the key to 1 to automatically enable ProGear Remote Software Update. Otherwise, set it to 0 to initiate the update manually.

RemoteUpdateServer

- Host name or IP of the server that has the ProGear Remote Software Update Service running

EnableToolboxRU

- When it is set to 1, the button to initiate ProGear Remote Software Update will appear in the Toolbox.

ManualUpdateReq
- When ProGear boots up, it checks this key to see if the user has requested the ProGear Remote Software Update

2.2. section DT_INPUT
US Keyboard
- Set to 1 for US onscreen keyboard layout

Keyboard Pics
- Directory of the onscreen keyboard map

HandwritingSetup
- The path of the setup application for the handwriting application

InputMode
- When the key value is 0, the default text input method is the onscreen keyboard. Otherwise, the handwriting application will be used.

2.3. section Dt_cleanup
CleanPluginCache
- When the key is set to 1, the plugin cache will be flushed during ProGear bootup

CleanCacheAndCookie
- When the key is set to 1, Netscape cache and cookie will be flushed during ProGear bootup

CleanBookmarksAndHistory
- When the key is set to 1, Netscape bookmark and history will be flushed during ProGear bootup

Example of a ProGear Registry

```
# DT Registry Data Version 1.04
*ProGear
HomeDirectory=/home/webzine
AppsDirectory=/usr/local/webzine/apps
Frontpath link=http://www.frontpath.com

*Screen
LCDwidth=1024
LCDheight=768
Width=1024
Height=768
Orientation=0

*Preference
Border=4
DefaultMenubarUp=1
DominantHand=1
VolumeMute=0
MasterVolume=99
PowerSavingMode=0
Brightness=100
*Network
SSID=webpad
EncryptionKey=
LoginID=Progear
EnableEncryption=0
EnableNTP=1
NTPServer=ntp0.cornell.edu
StatusSite=www.frontpath.com
AutoRemoteUpdate=0
RemoteUpdateServer=cruise.frontpath.com
EnableToolboxRU=0
ManualUpdateReq=0

*DT_INPUT
US Keyboard=1
Keyboard Pics=./pics/US
HandwritingSetup=/usr/local/webzine/apps/textinput/handwritingsetup
InputMode=0

*MenuBar Data
Landscape Width=1024
Landscape Height=45
Portrait Width=768
Portrait Height=45

*Utility Data
Landscape Width=1024
Landscape Height=212
Portrait Width=768
Portrait Height=212

*Window Data
0=Dt_menubar
1=Dt_toolbox
2=Dt_input
3=Netscape
4=Ts_calibrate
5=Dt_connect
6=Dt_battery
7=Dt_clock
8=Dt_advsetup
9=Dt_diagnostic

*Dt_menubar
Attributes=9

*Dt_toolbox
AppName=/usr/local/webzine/apps/dt_toolbox/dt_toolbox
Attributes=5
Activated=0
WindowID=0
*Dt_input
AppName=/usr/local/webzine/apps/dt_input/dt_input
Attributes=5
```

```
Activated=0
WindowID=0

*Netscape
Attributes=3
Landscape X=0
Landscape Y=0
Landscape Width=1024
Landscape Height=768
Portrait X=0
Portrait Y=0
Portrait Width=768
Portrait Height=1024

*Ts_calibrate
Attributes=32

*CAL4PT
Attributes=32

*Dt_connect
AppName=/usr/local/webzine/apps/dt_connect/dt_connect
Attributes=65
Activated=0
WindowID=0
# 0 is OFFLINE
# 1 is ONLINE
NetworkStatus=0

*Dt_battery
AppName=/usr/local/webzine/apps/dt_battery/dt_battery
Attributes=65
Activated=0
WindowID=0
# 0 is External Power
# 1 is Battery Power
PowerSource=0
# 0 is N/A or No Battery
# 1 is Discharging
# 2 is Charging
# 3 is Full
BatteryState=0
PowerRemaining=80

*Dt_clock
AppName=/usr/local/webzine/apps/dt_clock/dt_clock
Attributes=1
Activated=0
WindowID=0

*Dt_advsetup
```

```
Attributes=1
Activated=0

*Dt_diagnostic
Attributes=1
Activated=0

*Dt_cleanup
CleanPluginCache=1
RestoreRealConfig=1
CleanCacheAndCookies=1
CleanBookmarksAndHistory=0
```

6.6. Netscape Configuration for ProGear

ProGear ships with Netscape Navigator for Linux, which includes built-in support for JavaScript (v. 1.3, ECMA-252 compliant) and Java (JDK v. 1.1). Detailed specifications can be found on Netscape's web site home.netscape.com and technical support can be found at help.netscape.com.

Note—Netscape for Linux does not support functionality specific to the Microsoft Windows environment, such as ActiveX controls.

ProGear ships with the following browser plug-ins installed:

- Shockwave Flash 4.0 r1.2
- PDF Viewer 4.0.5

The browser is configured to use the directory /home/webzine as its working directory. Configuration and other browser support files are stored in /home/webzine/.netscape, and MIME types are stored in /home/webzine/.mime.types. The file preferences.js contains most of the configuration data for Netscape.

6.7. Programming Support

As previously stated in Section 6.1, the standard ProGear distribution supports wireless web access. If you install or develop additional software, including the kernel modifications, then you must test on both a development distribution and a standard distribution. The standard distribution may lack necessary libraries, which you will have to provide for your installation. If your application does not work on the standard distribution, the "ldd" utility in /usr/bin/ldd will help you to detect which libraries are needed by your application in the non-functioning environment, and where the files are located in the functioning environment.

On a development distribution, you will find the standard full development suite of tools from GNU, including C and C++ compilers, emacs, gdb, and info and man pages. If your use of the ProGear requires modification of the kernel, please note that the standard ProGear distribution uses "nuni" as the boot loader. You will need to recompile the nuni loader if you modify the kernel.

For more information on nuni, see http://www.ibiblio.org/pub/Linux/system/boot/loaders/!INDEX.html. It is also important to note that there is no keyboard available until the kernel loads the USB drivers, so it is impossible to select between multiple kernels at boot time.

A. Appendix

User Documentation

ProGear Getting Started Guide - http://www.frontpath.com/support/s_manual.htm
ProGear User's Guide - http://www.frontpath.com/support/s_manual.htm

Accessing the ProGear Development Environment (Errata Sheet)

Usernames and passwords:

- webzine / webZine
- root / webZine

Development Distribution

The development distribution is designed for operation with a keyboard attached and boots to a login prompt in multi-user mode. When logging in as root, you will be presented with a text interface and can start X-windows by typing "startx."

When logging in as webzine, type "startx." X-windows and the ProGear user environment are started automatically. To exit from the user environment:

- ➢ CTRL-DEL will log you out
- ➢ CTRL-ALT-<function key> will switch you to a different virtual terminal
- ➢ CTRL-ALT –F1 will connect you to the console terminal
- ➢ CTRL-ALT -F7 will connect you to an X virtual terminal

ProGear Distribution

The ProGear distribution boots into the user environment. To leave the environment:

- ➢ CTRL-ALT-F2 or CTRL-ALT-F3 will switch you to a different virtual terminal.

The user environment is designed to respond if an explicit kill or server shutdown is detected.
It cannot be escaped permanently.

Obtaining Miscellaneous Information

- Versioninfo will get you information on the current ProGear distribution
- ifconfig will get you the current IP address of the unit, in order to connect across a network for telnet, X-window, ftp, or nfs sessions

B. Appendix

User Interface Design Guidelines

This appendix describes design guidelines for the ProGear user interface. It also gives suggestions for application design, to provide for the development of a consistent look and feel and ease of use for ProGear-based interfaces.

Overview

These guidelines are intended for developers interested in designing ProGear-based interfaces. Developers will find this information useful for application design. This appendix is outlined as follows:

Section 1 - General Guidelines
This section provides design guidance beneficial to the user and the developer. The concepts discussed establish a framework for the interface onto which more specific guidance can be added.

Section 2 - Software Interface Conventions
This section discusses the software components of the ProGear interface.

Section 3 – User Interface Controls and Components
This section discusses the controls and components used to access the interface and the system utilities. The capabilities and limitations of the components are discussed along with suggestions to aid application design.

B1. General Guidelines

Understand the User

Consider the user's needs when making design decisions. Consider the goals of the user.
What is the user trying to accomplish? Always design the interface around the critical and frequent features first, then add other features.

Be Consistent

Be consistent with display locations and control elements. Consistency allows the user to concentrate on the task, rather than on the interface. Once a concept is learned, it can be applied throughout the application.

Keep the User Informed

Keep the user informed and provide immediate feedback. Error messages and documentation should be geared to the user, not the programmer. Use standard message areas and popups for message displays and make sure that the message is erased when the next operation is started.

B2. Software Interface Conventions

B2.1. Splash Screen

During the boot up process, ProGear displays a graphical splash screen that shows the frontpath and ProGear names and logos. The splash screen will include copyright information and indicate that the device is initiating. During the process of powering down or shutting down, ProGear displays a screen that indicates the unit is turning off or shutting down. No error messages, script commands, or comments will be visible on the screen during the boot up or power down.

B2.2. Applications

The Netscape Navigator browser is the only application resident on the unit along with several system utilities. Access to the interface requires a stylus or a fingertip to activate the touch screen and invoke the system utilities.

Web Browser

The current browser is the Netscape Navigator (ver. 4.7x or newer) browser for the Linux operating system. Under user control, this browser can be viewed in either portrait or landscape orientation. The default orientation is landscape. frontPath has made the following changes to the user interface of this product:

- The default Home page is specified by frontPath or its partner
- A core set of Bookmarks is specified by frontPath and its partner
- The Bookmarks list may include entries previously specified by the user
- The initial interface configuration includes fonts selected for optimal readability on the ProGear

Multiple instances of the browser are possible. The system will create an initial browser instance at power on. At least one browser instance remains active throughout the user session. If a system failure occurs forcing a system reboot, when the system restarts, it will reestablish the browser window and, if possible, restore the user to the last page viewed.

There are controls to allow the user to create new instances of the browser, switch among the various browser instances, and close an instance of the browser. The user cannot, however, be able to close the last active browser.

For web-based applications, the user should always be in the browser. ProGear’s operating system limits the user’s access to only the browser system utilities and the GUI environment. This ensures simplicity and ease of use. As a web-based application developer you do not have to worry about users leaving the operating system.

B2.3. Utilities

There is set of system utility applications resident within the unit. The user can invoke a single utility at a time, with each appearing in the same window successively. The utility window includes a control to allow the user to close the window. This dedicated utility window shares screen space with the browser window but does not obstruct it. When the user closes the utility window, the system will restore the screen space it occupied to the browser. During the time a utility is active, application and browser windows are resized.

B3. User Interface Controls and Components

B3.1. Menubar

The Menubar contains buttons to help the user navigate, and icons that provide information about the system. The Menubar provides a quick way for users to get to frequently used commands. Keep this in mind when deciding what commands should be put in the Menubar. The space across the bottom of the screen is reserved for the Menubar. This area extends ½" to 1" above the bottom of the screen. System icons, tools, and status indicators populate this area. The Toolbar is visible at all times. Once an item in the Toolbar is selected, an audio cue will sound and the cursor changes to a watch or hourglass while the selected item is activated or opened.

From left to right on the Menubar, you see the following:

1. frontpath logo button - sends the browser to a locally resident HTML page. This is the ProGear Support page.
2. Icon to activate the Toolbox Panel. This panel allows the user to change basic and advanced system settings.
3. Icon to activate either the onscreen Keyboard or Handwriting Recognition modules, depending upon which was used last. The onscreen Keyboard is the factory default setting. A keyboard icon represents the onscreen Keyboard. The Handwriting Recognition module is represented with an icon that depicts writing.
4. Netscape icon to activate new instances of the web browser.
5. Status indicator for the Network Connection Status. This icon will indicate if a connection exists between the ProGear and the wireless access point. If no connection exists, the icon will display a red circle and slash. Tapping the Network Connection Status indicator activates a window with detailed connection information, including SSID and encryption settings, connection status, and connection signal strength (providing the wireless card supports these features).
6. Status indicator for the Audio Status. This icon indicates whether the audio is currently muted or played. If the audio is muted, the icon displays a red circle and slash. Tapping the icon toggles between muted and unmuted audio.
7. Status indicator for the S.M.A.R.T Battery. It displays the battery with AC cord icon when connected to AC power, and displays the battery icon only when not connected to AC power. The icon displays battery charge status by number and color of the bars in the icon. Green bars indicate a good charge; yellow bars indicate low power; a red bar indicates that the system needs to be connected to AC power.
8. Status Indicator of Time and Date. Displays the time according to the Time Server. The time is displayed in standard 12 hour AM/PM mode. The time will be based on the current system value, which is set in

the Toolbox Panel either manually or via a Network Time Server. Tap the Time icon to activate a window that displays a calendar. The calendar allows you to view the month, day, and year.

9. Desktop Manager Controls. The Desktop Manager allows the user to control which window or application is active and on top. Controls also allow the user to Close the selected window or to change the active window to the next window currently open. The Desktop Manager includes:

- ♦ Close button to close the active window.
- ♦ Pop-Up list of currently open windows and applications. Controls which open window or application is active and on top
- ♦ ZZZ button. Pressing this button puts ProGear into Standby mode.

B3.2. Toolbox

ProGear comes with a set of system utility applications and settings accessible from the main Toolbox window. Advanced or restricted access utilities and settings are available from the Advanced Toolbox button. The user can select the Advanced Toolbox from the standard Toolbox window. All utility windows include a control to allow the user to close the window. When selecting the Advanced Toolbox, the standard Toolbox window will close and be replaced with the Advanced Toolbox window. The user must enter the correct password to access the Advanced Toolbox window.

This Toolbox window shares screen space with the browser window but does not occlude it. When the user closes the Toolbox window, the system will restore the screen space it occupied to the browser. The basic Toolbox panel includes the following settings and controls:

Screen Brightness
This slider allows the user to adjust the brightness of the LCD display.

Touch Screen Calibration
This button activates a utility to allow the user to calibrate the touch screen for pen input.

Advanced Settings
This button activates the Advanced Toolbox utility. Selecting this button will open a window requesting the Advanced Toolbox password. If the user enters the correct password, then the basic Toolbox window closes and the Advanced Toolbox window opens. If the wrong password is entered, the system displays a message that the password is incorrect and the system will beep. Then the password window will disappear, returning the user to the main Toolbox window.

Power Saving Modes
The Power Saving Mode settings control the system’s power saving modes. The settings are activated in the Toolbox. The user can select from the following choices: Disable, Minimal, Average, and Aggressive. The current system settings are available in the Profile. Power saving modes can be selected for the period of inactivity that triggers the LCD brightness level to be reduced, the period of inactivity that triggers the system to go into Standby mode, and the period of inactivity that triggers the system to go into Off mode.

Screen Rotation
These buttons allow the user to change the orientation of ProGear to portrait-left, portrait-right, or landscape.

Right- and Left- Handed Buttons
These buttons set the user preference for right or left -handed input. Text input is adjusted based on these preferences. Only one button is active at a time. The other button is deactivated.

Audio Settings
The Mute button allows the user to toggle between audio muted and unmuted settings. The Audio Out Volume slider allows adjustment to the audio volume level for the audio out and built-in speaker.

Diagnostics
This button activates the built-in system diagnostics utility. This utility displays the system and extended network configuration and performs system tests.

Manual CRUiSe Update Button
Selecting this button sets a flag within the OS to run the Remote Software Update utility at the next appropriate opportunity. This will happen the next system restarts. If the remote software update finds applicable updates for the ProGear, the updates will be downloaded and installed. After completing the installation, the remote software update will restart the system. The remote software update will then create a dialog box to notify the user that the system is automatically checking for updates. If an update is found, the dialog box notifies the user that the system is installing updates and that the system will restart upon completion.

Note—ProGear must be restarted to run the automatic or manual remote update. The user must also have ProGear connected via the AC/DC power adapter to run the remote update.

B3.3. Advanced Toolbox Settings

The Advanced Toolbox utility is password protected. The Advanced Toolbox includes the following settings, controls, and utilities:

Change Advanced Toolbox Password
The user is able to change the password required to enter the Advanced Toolbox utility. The factory default password is “Progear” (the password is case-sensitive). When entering the new password, the user must enter it twice. The actual characters typed are not displayed on the screen.

Network Name/SSID Settings
The user can enter the desired SSID name. The Service Set ID (SSID) is the name of the access point that ProGear should communicate with. The name used in ProGear should match the name of the access point.

Wireless Encryption Settings
Encryption techniques are used to safeguard information while it is stored within a network node or while it is in transit across communications media between nodes. The user can select Encryption on or off. If Encryption is turned on, the user must enter the Encryption Key used on their network.

Note—The encryption setting selected must be supported by both the network card and access point.

Time and Calendar Settings
The user can select whether to use a Network Time Server or manual entry to set the time and date. If a Network Time Server is used, the user will be able to enter the server name or IP address. A

Synchronization button allows the user to manually request the ProGear to resynchronize with the Network Time Server.

B3.4. System Utilities

Text Input

The text input utility enables the user to enter computer-readable text using the stylus. The utilities include an onscreen keyboard, hot keys module, and a handwriting recognition module. These utilities provide access to all alphanumeric characters of the ISO Western character set.

Onscreen Keyboard Module

The keyboard enables user entry of single characters using a graphical, onscreen keyboard. The keyboard appears in a separate window that can be moved around the screen. The keyboard should not occlude the Menubar.

The keyboard fits on the screen in portrait or landscape modes. It contains the standard QWERTY keyboard layout, plus the following additional keys: ESC, Backspace, Tab, Delete (forward delete), Caps Lock, Enter, Shift, Control, Alt, Space, Left/Right and Up/Down Arrows. The arrow keys must be able to control the text cursor's movement in text fields.

The keyboard appears quickly on the screen once it is selected from the Menubar. The keys make an audible click when depressed. It also includes a button to switch to the Handwriting Recognition module.

Handwriting Recognition Module

The handwriting recognition module includes a method for the user to enter a single, handwritten character or an entire word. Words entered by the user are compared against an internal dictionary with the best results presented to the user. The user can select the appropriate match.

The text input utility window does not occlude the currently visible browser window. Instead, the system resizes the currently visible browser window when the user invokes the text input application, and then resizes the browser window again when the user closes the text input application window. Text entered via word or character entry will only appear in the current, active text field selected by the user.

The handwriting recognition module includes buttons for Left and right Tabs, Backspace, Enter, Undo, Space, Autospace, and Setting. For character entry, the user must select uppercase, lowercase, numbers, or symbols prior to entering a character.

The Setting button will bring up the handwriting settings window. In this window, the user can adjust settings for:

- How long the software must wait after a user stops writing before analyzing the word
- Whether or not to automatically insert a space after each word
- What is the minimum confidence percentage for including words in the suggested list
- Whether or not to automatically choose the best guess from the suggested list when the top guess is above 95%.

Desktop Manager Controls

The user can select from a list of open applications and windows to determine the active window. To help users activate your windows, always assign descriptive titles to all windows and dialog boxes. Also keep window titles as short as possible. The Next button allows the user to quickly change between open windows and applications. The Close button will close the instance currently selected in the Desktop Manager list.

User Notification

There shall be methods for the system to notify the user of various conditions that may adversely affect usability. These notifications may include:

Alerts
There are system messages, viewable by the user, which report conditions that will affect the user but require no user intervention. Among these conditions are:

- Low battery
- Imminent system shutdown
- Low memory
- Loss of the network connection
- System crash, reboot in progress

Status Indicators
Included are interface elements, visible to the user, which the system uses to inform the user of current system activity or status. Among these activities and states are:

- Audio output muted/unmuted
- Battery charge remaining
- Battery charging
- Current active window/instance
- Wireless network connection
- Time

Audio Cues
There are distinctive audio tones or tone sequences, audible to the user through both the monaural speaker and the stereophonic headphones, which the system will use to indicate certain activities and occurrences to the user. Among these activities and occurrences are:

- Keyboard clicks (for onscreen keyboard)
- System power off
- System shut down
- System start
- Toolbar clicks (click sounds when user taps Toolbar icons)
- User input error

B3.5. Text Input Design Considerations

When designing for the text input interface, consider the following:

- Because ProGear can use an XGA screen display, text type can be larger for readability. Please be advised that units can ship with XGA or SVGA displays.
- Since ProGear will often be used without a physical keyboard, whenever possible, provide stylus-based methods to input data instead of using a keyboard (e.g., buttons, popup menus, pull-down menus).

When designing pull-down menus, keep in mind the following:

- Keep menu names to one word
- Keep two menu levels or less if possible
- List items in order of use

When designing popup menus, keep in mind the following:

- Popup menus should appear centered on the main window
- Try to avoid displaying a series of popup menus, allow user to get information from one popup

C. Appendix

Sample Script Files

C1. Manifest File Script

A script to create a new manifest file. Filename: mkupdate

```
#!/bin/bash

if [ "$1" = "" ]; then
   echo
   echo "Usage: " `basename $0` " <update file>.tgz"
   echo
   echo "   Note: the .tgz extension will be automatically appended."
   exit
fi

# Remove old manifest file if it exists.
if [ -f CRUiSe/CRUiSeManifest.txt ]; then
   echo "Removing Old Manifest File..."
   rm -f CRUiSe/CRUiSeManifest.txt
fi

# change to CRUiSe directory where the files for the update
# reside.
cd CRUiSe

# Create a new manifest file with MD5 checksums
echo "Creating New Manifest File..."

find * -type f -exec md5sum {} \; > CRUiSeManifest.txt

# Return to the original directory
cd ..

# if an update of the same name exists, delete it.
if [ -f $1 ]; then
   echo "Removing Old Update File $1"
   rm -f "$1.tgz"
fi

# Now create the new compressed tar file that is the "update"
echo "Creating New Update File: $1"
tar -cvzf "$1.tgz" CRUiSe

# now copy it to my server directory
cp "$1.tgz" /home/httpd/cgi-bin/update.tgz
```

C2. Remote Software Update Manifest

A sample manifest file for a remote software update package. Filename: CRUiSeManifest

```
28e92c63bf559882ddab405019f492a3     CRUiSeUpdate.sh
b148bc99b3c59ef56c7b71a184b15861     _emacs
5ec59b9c05706b4ce65adf44d0d3ab24     ls
```

```
96f4e0549cf53845f65af76867d91dd4     md5sum
c0ce4353bf86c71124f8c2cb3df5d93e     subdir/mkdir
72fdc0385f84c11f96f4b332dc1fe48d     subdir/mknod
0f54aadf84c69adb67d7a45951f985ad     subdir/mktemp
```

Note—The column references are: MD5 checksums, one or more characters of white space, and filenames.

C3. Remote Software Update Shell Script

A sample shell script file. Filename: CRUiSeUpdate.sh

```
#!/bin/bash

echo "Hello World!"
date > /tmp/blah

./ls -l
./md5sum ls

mv /home/garysa/devel/software/cruise/cruise
/home/garysa/devel/software/cruise/cruise_OLD
cp /home/garysa/devel/software/cruise/cruise_no_debug_info
/home/garysa/devel/software/cruise/cruise

exit 21
```

C4. Perl Script

Sample of a simple remote software update server Perl script. Filename: CRUiSe.pl

```
#!/usr/bin/perl
use MD5;

$CRUISE_DO_NOTHING_MODE   = 0;
$CRUISE_FINISHED          = 1;
$CRUISE_CHECK_WITH_SERVER = 2;
$CRUISE_UPDATE_MODE       = 3;
$CRUISE_HDSCAN_MODE       = 4;


$CRUISE_ERROR_OCCURRED    = 5;
$CRUISE_GUID_GEN_REQUEST  = 6;
$CRUISE_UPDATE_CONF_FILE  = 7;

$SERVER_MODE = $CRUISE_DO_NOTHING_MODE;

#print( STDERR "Request Method = $ENV{'REQUEST_METHOD'}\n" );
#print( STDERR "Content Length = $ENV{'CONTENT_LENGTH'}\n" );

#print( STDERR "Hey, Gary! - ", scalar( localtime ), "\n" );

if ( $ENV{'REQUEST_METHOD'} eq "\UPOST" )
{
     while ( $line = <STDIN> )
     {
    chomp($line);
    if ( $line =~ /X-FrontPath-(.*):\s+(.*)$/io )
    {

             #$damit .= "$line\n";
        $clientMsgs{lc($1)} = lc($2);
        #print( STDERR "(CRUISE) $line\n" );
    }
    elsif ( $line )
    {
        #print( "$line\n" );
        push( @dataLines, $line );
    }
     }
}

#$damit .= "X-FP-Mac: $clientMsgs{mac}\n\n";

#print( STDERR ">> clientmode = $clientMsgs{'clientmode'}
($CRUISE_CHECK_WITH_SERVER)\n" );

if ( lc( $clientMsgs{'mac'} ) eq "00:90:4b:00:6d:a8" ||
     lc( $clientMsgs{'mac'} ) eq "00:90:4b:00:6e:79" ||
     lc( $clientMsgs{'mac'} ) eq "00:90:4b:00:6d:b9" ||
     lc( $clientMsgs{'mac'} ) eq "00:90:4b:00:6e:00" ||
     lc( $clientMsgs{'mac'} ) eq "00:90:4b:00:6e:3e"
   )
{

        $MODE = $CRUISE_UPDATE_MODE;
}
elsif ( $clientMsgs{'clientmode'} == $CRUISE_CHECK_WITH_SERVER )
{
    #$MODE = $CRUISE_HDSCAN_MODE;
    $MODE = $CRUISE_UPDATE_MODE;
   #$MODE = $CRUISE_DO_NOTHING_MODE;
}

elsif ( $clientMsgs{'clientmode'} == $CRUISE_HDSCAN_MODE )
{
     if ( @dataLines )
     {
    #print( STDERR "Bad Files From $clientMsgs{'mac'}...\n" );
    foreach $line ( @dataLines )
    {
        #print( STDERR "$line\n" );
    }
     }
     $MODE = $CRUISE_FINISHED;
}
elsif ( $clientMsgs{'clientmode'} == $CRUISE_UPDATE_MODE )
{
     if ( @dataLines )
     {
    #print( STDERR "Output From $clientMsgs{'mac'}...\n" );
    foreach $line ( @dataLines )
    {
        #print( STDERR "$line\n" );
    }
     }
     $MODE = $CRUISE_FINISHED;
```

```
}
elsif ( $clientMsgs{'clientmode'} == $CRUISE_GUID_GEN_REQUEST )
{
    $MODE = $CRUISE_UPDATE_CONF_FILE;
}

##################################################################################
##################################################################################

if ( $MODE <= $CRUISE_CHECK_WITH_SERVER )
{
    $UPDATEFILE = "";
    $CONTENT_TYPE = "frontpath/nada";
    $size = '0';
}
elsif ( $MODE == $CRUISE_UPDATE_MODE )
{
    #------------------------------------------------------------
    # Change $UPDATEFILE to correspond to the patch
    # you want sent to the pad.
    #------------------------------------------------------------

    #$UPDATEFILE = "update-20326.tgz";
    $UPDATEFILE = "update-22011.tgz";


    #$UPDATEFILE = "cruise-20326.tgz";


    #$UPDATEFILE = "cruise-cal2.tgz";

    # do not change this line!
    $CONTENT_TYPE = "frontpath/update";
}
elsif ( $MODE == $CRUISE_HDSCAN_MODE )
{
    $UPDATEFILE = "md5list.txt";
    $CONTENT_TYPE = "frontpath/scanrequest";
}
elsif ( $MODE == $CRUISE_UPDATE_CONF_FILE )
{
    $UPDATEFILE = "";
    $CONTENT_TYPE = "frontpath/updconf";

    $GaryzGUID = &genGUID();

    $NewConf = ( "CruiseHostName: \"cruise.frontpath.com\"\n".
           "CruisePort: \"80\"\n".
#          "CruisePadSWRev: \"0\"\n".
           "CruisePadID: \"$GaryzGUID\"\n" );

    $md5    = new MD5;
    $md5->add( $NewConf );
    $digest = $md5->digest();
    $MD5    = unpack("H*", $digest); # $md5->hexdigest();
    $size   = length( $NewConf );
    #print( STDERR "Digest is " . unpack("H*", $digest) . "\nLength Is " . $size .
"\n" );

}
else
{
    DIE;
}
```

```
if ( $UPDATEFILE ne "" )
{
     ($MD5) = split( " ", `md5sum $UPDATEFILE` );

     ($dev,$ino,$mode,$nlink,$uid,$gid,$rdev,$size,
      $atime,$mtime,$ctime,$blksize,$blocks)
    = stat($UPDATEFILE);
}


#X-FrontPath-FileName: $UPDATEFILE

syswrite( stdout, "X-FrontPath-UpdateMode: $MODE
X-FrontPath-MD5CheckSum: ${MD5}
X-FrontPath-ContentLength: $size

Content-type: text/html
$damit
" );

if ( $UPDATEFILE ne "" )
{
     open( infile, $UPDATEFILE ) || die;
     while ( $x = read( infile, $buf, 1024 ) )
     {
    syswrite( stdout, $buf, $x );
     }
}
elsif ( $NewConf )
{
     #print( $NewConf );
}

#print( STDERR "Buh Bye, Gary! - ", scalar( localtime ), "\n",
#        ( $MODE == $CRUISE_FINISHED )? "\n" : '' );


sub genGUID
{
     my( $p0, $p1, $p2 );
     $p0 = $clientMsgs{'mac'};
     $p0 =~ s/://g;
     $p1 = `cat /proc/[a-z]* 2>&1 |md5sum`;
     ($p1) = split( " ", $p1 );
     $p2 = sprintf( "%08x", $$ * rand(123456) );
     return( uc("$p0-$p2-$p1") );
}
```

D. Appendix

Current Software Release Notes and Errata

See the developers support site for the latest software release notes and errata at http://www.frontpath.com/support.

Other Support Resources

For ProGear developer's information, sign up for the Developer's Email News List. Please send your request to: Devinfo@frontpath.com.

If you are interested in co-marketing opportunities or strategic partnerships with frontpath, please send a brief description of your proposal and contact information to:
Bizdev@frontpath.com.

frontpath, Inc.
2841 Mission College Blvd.
Santa Clara, CA. 95054
408-588-8800
http://www.frontpath.com

Bibliography

Linux Device Drivers - O'Reilly & Associates, Inc., by Alessandro Rubini

The Cathedral & The Bazaar - O'Reilly & Associates, Inc., by Eric S. Raymond, ISBN 1-56592-724-9, 288 pages, October 1999

Linux in a Nutshell, 3rd Edition - O'Reilly & Associates, Inc., by Ellen Siever, Jessica P. Hekman & Stephen Figgins, ISBN 0-596-00025-1, 650 pages, July 2000

Linux Network Administrator's Guide, 2nd Edition - O'Reilly & Associates, Inc., by Olaf Kirch & Terry Dawson, ISBN 1-56592-400-2, 450 pages, July 2000

Understanding the Linux Kernel - O'Reilly & Associates, Inc., by Daniel P. Bovet & Marco Cesati, ISBN 0-596-00002-2, 650 pages, September 2000

Vendor Datasheets

TBD