Phindows Connectivity
User's Guide

QNX SOFTWARE SYSTEMS

Published under license by:

QNX SOFTWARE SYSTEMS

QNX Software Systems International Corporation
175 Terence Matthews Crescent
Kanata, Ontario
K2M 1W8
Canada

P: +1 613 591-0931
F: +1 613 591-3579

info@qnx.com
http://www.qnx.com

Publishing history:	
First Edition	Fall 2009

Date of publication: Tuesday, March 23, 2010

Table of Contents

Preface
Typographical conventions

Throughout this manual, we use certain typographical conventions to distinguish technical terms. In general, the conventions we use conform to those found in IEEE POSIX publications. The following table summarizes our conventions:

Reference	Example
Code examples	`if( stream == NULL )`
Command options	`-lR`
Commands	`make`
Environment variables	PATH
File and pathnames	`/dev/null`
Function names	*exit()*
Keyboard chords	`Ctrl-Alt-Delete`
Keyboard input	`Username`
Keyboard keys	`Enter`
Program output	`login:`
Variable names	*stdin*
Parameters	parm1
User-interface components	**Navigator**
Window title	**Options**

We use an arrow in directions for accessing menu items, like this:

You'll find the Other... menu item under **Perspective → Show View**.

We use notes, cautions, and warnings to highlight important messages:

Notes point out something important or useful.

Cautions tell you about commands or procedures that may have unwanted or undesirable side effects.

Warnings tell you about commands or procedures that could be dangerous to your files, your hardware, or even yourself.

Note to Windows users

In our documentation, we use a forward slash (/) as a delimiter in all pathnames, including those pointing to Windows files. We also generally follow POSIX/UNIX filesystem conventions.

Preface
Technical support

Technical assistance is available for all supported QNX products.

To obtain technical support for any QNX product, visit the Support + Services area on our website (*www.qnx.com*). You'll find a wide range of support options, including community forums.

Chapter 1
How to Use This Guide

This *User's Guide* describes the how to use Phindows connectivity to connect to a remote system. This guide contains getting started information, as well as quick command line and configuration setting reference.

The following table may help you find information quickly:

To:	Go to:
Get up and running using Phindows	*Getting started with Phindows* (p. 3)
Use the command line options	*Configuring Phindows* (p. 29)
Configure Phindows	*Using the command line options* (p. 11)

Chapter 2
Getting started with Phindows

Phindows is a connectivity tool that lets you use Microsoft Windows platforms to connect to and interact with graphical Photon applications running on a remote Neutrino computer.

`dtype`

envar

This chapter describes how to get started using Phindows.

Starting Phindows

When you start Phindows, it displays a Connect dialog where you can specify the type of connection (TCP/IP or direct-connect serial), compression type, cache settings, and whether to connect to an existing Photon session. Various connection options are available, but the defaults usually work well.

Once you've chosen the connection parameters, click **Connect** to cause Phindows to connect with these parameters. To record these settings permanently before making the connection, click **Save**. These saved settings become the defaults for all future Phindows sessions.

If you check **Connect to Existing Photon Session** and select **List**, you'll be prompted with a list of existing sessions on the target machine. These sessions may include the main `/dev/photon`, and any other sessions created by `phrelay`. Private photon sessions keep running if a `Phindows` connection is unexpectedly terminated (if `phrelay`, `Phindows` or the network connection crashes, for example), or if you choose to not terminate the session when exiting `Phindows` normally. Active Photon sessions are listed in `/dev`, named *ph* followed by the process number of the `phrelay` that created them.

If you request a TCP/IP connection, you must also specify the Internet address of the Neutrino computer you're connecting to (e.g. *198.53.31.1*), or the hostname. If the remote computer has been configured properly, you should see a Photon login prompt, at which point you're connected and running Photon.

If you request a serial connection, you must specify the COM port (e.g. COM1 or COM2). If you don't specify a baud rate, Phindows uses the current Windows default settings. With a serial connection, Phindows initially acts as a simple text terminal that lets you type commands directly to the modem (e.g. ATDT1-613-591-0934). Once connected, log into Neutrino and then issue the command:

On QNX 4:

```
/usr/photon/bin/phrelay
```

On QNX Neutrino:

```
/usr/bin/phrelay -x
```

This command causes Phindows to drop out of text-terminal mode and begin acting as a Photon graphical terminal. A Photon login screen should appear at this point.

Configuring Phindows for TCP/IP use

If you're using TCP/IP, there are several configuration issues that you need to check before you can use Phindows. The configuration should already be set correctly; if not, make the changes described below.

The remote QNX host must have TCP/IP installed and running. In addition, `etc/inetd` must be running with the following additional items added.

To configure Phindows for use with TCP/IP:

1. In `etc/inetd.conf`,add the following line:
   a) On QNX 4:`phrelay stream tcp nowait root /usr/photon/bin/phrelay phrelay`
   b) On QNX Neutrino:`phrelay stream tcp nowait root /usr/bin/phrelay phrelay -x`
2. In `/etc/services`, add the line:`phrelay 4868/tcp`

These lines are already present in the configuration files, but they're commented out. Just remove the number sign (#) to add these entries.

These two entries cause `inetd` to listen for incoming requests to establish a new Photon session. When a request is detected (from a remote Phindows client in our case), `inetd` automatically establishes a full TCP/IP connection and launches `phrelay` on that connection. Phindows is then fully connected to the local machine.

For more information about `inetd`, `inetd.conf`, and `phrelay` see the QNX Neutrino *Utilities Reference*.

All that remains now is to configure the Windows client to support TCP/IP. The exact way to do this depends on the type of Windows platform you're using, what services have already been installed on that client, and whether you're using Ethernet or dialup networking over SLIP/PPP (see the Microsoft documentation for help with this).

Data-compression options

Phindows uses data compression to compress the data that is sent from host computer to QNX target.

You can use the Phindows Connect dialog to specify one of these data-compression options:

Byte-Pair Encoding (BPE)
Performs the best data compression and is recommended for all serial connections. Although BPE is inexpensive to decode (both in memory and CPU usage), it does require a fair amount of CPU on the host QNX machine. When link speeds are low (such as with modem links), the extra cost in CPU on the sending side is well worth the tradeoff.

Run Length Limited (RLL)
Provides good data compression. Although not as highly compressed as BPE, RLL encoding is both quick to decode and encode. We recommend RLL encoding for network TCP/IP connections because it places less CPU load on the QNX host, yet provides fair compression.

Gzip Image Compression
Provides excellent compression using a modified Lempel-Ziv algorithm (LZW). You can enable Gzip image compressoin by selecting **Gzip Image Compression** on the **Connect** dialog.

None
With data compression turned off, Photon draw events are transmitted as-is. This might make sense if the connection bandwidth is VERY high (say 100Mb Ethernet), making the extra processing time involved in data compression an unnecessary step.

Data-caching options

Phindows makes very extensive use of data caching techniques to minimize the amount of data that needs to be transmiDatacachingtted from the host to the client machine. Thanks to this intensive data caching, Phindows is usable even on modem-link speeds as low as 9600 baud.

In addition, Phindows and `phrelay` can optionally compress cache files (larger than 63,000 bytes) using the `gzip` utility to further improve performance. See the -z command-line option below.

Most large static data objects in Photon are tagged with a unique 32-bit ID, or *tag*. Tagged objects include bitmap data, image data, and color palettes. These objects are generally tagged when first created by the Photon program developers. This process is accomplished automatically by the Photon development tools, so the program developer is, for the most part, not even aware that this is taking place.

Photon passes these tags along with the data objects as they flow through the Photon event space. The end result is that Photon graphics drivers (such as `phrelay`) almost always have available a small, unique, precalculated tag to identify the larger Photon graphical objects.

The host side of a Phindows connection, `phrelay`, has the job of transmitting the graphical Photon events over a communications link to a Phindows client. Phindows and `phrelay` use a private protocol that allows these large graphical objects to be sent only once to the remote client. All subsequent references to that data object are made by passing the small tag, knowing that the Phindows client has a local copy of the entire data object. Over time, commonly encountered objects (such as icons and backdrops) all become cached.

Phindows not only caches these tagged data objects in local memory, but also spills least-recently used objects to disk automatically and saves all cached objects to disk when you close Phindows. The next time that you start Phindows, it preloads its in-memory cache from the most recently encountered disk cache and informs the remote `phrelay` session which objects it already knows about. This means that after a graphical object is seen once within Phindows, it never, in general, has to be transmitted again, even in subsequent Phindows sessions.

The Phindows Connect dialog lets you tune the size of the in-memory data cache and the maximum amount of disk storage to reserve for most recently encountered objects. The default values of 16384K for RAM cache and 80M for disk cache are usually adequate.

Using Windows fonts

In normal operation when connected to a remote host, phindows receives all text as bitmaps which have already been rendered by the Neutrino font manager. This mode of operation is best in most situations, but at very slow connection speeds you can improve performance by configuring Phindows to use local Windows fonts.

Phindows uses local Windows `.ttf` (TrueType) and `.phf` fonts rather than font bitmaps rendered by `phrelay`, if it finds them in its font directory. This feature is available in `phrelay` 6.3 or greater and Phindows 2.13 or greater.

To use Windows fonts:

1. Copy the `.phf` and `.ttf` font files to the `font/` subdirectory used by Phindows.

   To determine the location of the `font` subdirectory, on the **Connect Dialog**, click **More >>**. The font subdirectory is located in the file location that is specified by the **Cache Dir:** field. Place any fonts in this subdirectory in order to use them in your Phindows session.

2. Install the copied `.ttf` fonts in Windows using the Fonts control panel. This procedure depends on the version of Windows you are running. See the Windows online help for more information on installing fonts.

You can check which fonts are being picked up from the Neutrino or Windows side by starting `phrelay` with the `-V` and `-D` command-line options. For example, you can change the `/etc/inetd.conf` file to start `phrelay` as follows:

```
phrelay -x -V -Ddbg_log.txt
```

Then restart `phrelay` (for example, with `slay -s HUP inetd`) to get debug output sent to `/dbg_log.txt`. With the `-V`, debug lines for all text rendered using a font present at the Phindows end are recorded. To see which fonts are still being rendered by `phrelay` and sent as bitmaps, use the `-VVV` option, which produces more verbose debug information. In the resulting debug output look for the string *render opts*, which is output every time `phrelay` calls `PfRenderText()`. The line before this one gives the font filename, and a few lines after is an ASCII representation of the rendered text bitmap.

Draw buffer size errors

When connecting to a machine running an older version of `phrelay`, it's possible to exceed the draw buffer limit. You can use the Windows font rendering option to reduce the draw buffer size.

Using snapshot or PgReadScreen()

Phindows supports taking screenshots with the snapshot utility, or any other application that calls PgReadScreen().

When Phindows is connected to an existing Photon session (using -n), the screenshot is read directly from the original display without any special processing.

When Phindows is connected to a private Photon session (-n isn't specified), a copy of the screen contents inside the Phindows window is sent from Phindows to the Neutrino machine via phrelay. If the link speed is slow and the screen contents are large and complex, this may take some time to complete. Watch for the status messages Phindows displays in its title bar for an indication of send progress. The screen data is compressed according to the same compression settings used for draw data received by Phindows. The BPE setting often results in better compression ratios than RLL, but requires significantly more CPU resources to calculate. (Gzip compression is not implemented for screen capture data.)

Screen captures gathered by Phindows are 24 bits-per-pixel images, no matter what the local and remote graphics settings are. If the Phindows double buffering feature is not turned on and you are not using Windows Vista or Windows 7, the Phindows window must be positioned correctly and can't be covered by any other windows.

Chapter 3
Using the command line options

Phindows uses command line options to define user settings and configuration options.

You can use the command line options to control settings such as connection speed, window size and look and feel, and sharing options.

Command-line options

`PHINDOWS.EXE` supports several command-line options.

You typically add these parameters to the command when you create the icon or shortcut to launch Phindows. The options are:

-b

***link_speed*[,*commbaud*]**

The effective link speed.

If you use a direct serial connection, then the comm port's baud rate is set to this value as well, unless you give a specific baud rate *(commbaud)*. If you don't specify a baud rate, the Windows default baud rate is used.

The effective link speed is used by Photon to decide if certain draw operations should be performed in a more simple bandwidth-saving way. This option allows you to specify a value independent of the serial port baud rate, if you wish. The effective link speed is also used for TCP/IP connections, in which case it defaults to 10000000 (10Mbps). If you have a faster network or don't want Photon to speed-optimize any draw operations (such as divider column resizing) then specify 100000000, which is 100Mbps. &"

-d *method*

Double Buffering method:

- 0 — disable
- 1 — use default method
- 2 — use Direct3D
- 3 — use DirectDraw
- 4 — use system memory

Introduced in Phindows version 3, double buffering greatly improves the speed at which portions of the Phindows main window get redrawn after they are damaged.

When double buffering is enabled, all draws received from the remote QNX system will be performed in an offscreen memory context and then blitted to the main window. This blitting can affect the performance and CPU usage of Phindows, so a number of blitting methods have been made available for the user to choose from. In theory, because of the hardware acceleration features of most modern video cards, blitting

within video memory should be faster than blitting from system memory into video memory.

Phindows can use two different methods to access video ram (Direct3D and DirectDraw), or it can blit using system memory. The specific software/hardware configuration of your computer will determine which method is faster. Direct3D is preferred, because it offers the most flexibility for future enhancements. To use this method you must have DirectX 9.0c (or later) installed on your computer. If the correct version of DirectX is not available, or if there is a failure initializing Direct3D, Phindows will fall back to using an older DirectDraw interface. If DirectDraw fails to initialize, or if it can't get access to video ram, then double buffering will be disabled. In order to enable double buffering with no video ram available, you must specify the "system memory" option.

The double buffering method choices are described in greater detail below:

Disabled

No double buffering will be performed. Performance will be the same as in older versions of Phindows.

Default

On Windows NT/2000/XP/Vista, Phindows will attempt to use Direct3D, falling back to DirectDraw if necessary. If both methods fail, double buffering will be disabled. On Windows 98/Me, double buffering will be disabled. On Windows Vista and Windows 7, double buffering will be disabled if Aero composition is active, otherwise the behaviour will be the same as on XP.

Direct3D

Direct3D will be used in all versions of Windows. If this fails, Phindows will notify the user, then fall back to using DirectDraw. If both methods fail, double buffering will be done using system memory.

DirectDraw

Direct3D will not be initialized. Phindows will attempt to use DirectDraw. If this fails, Phindows will notify the user, and then double buffering will be done using system memory.

System Memory

Direct3D and DirectDraw will not be initialized. Double buffering will be done using regular system memory. In some cases this method can perform better than using video memory.

Windows Vista (Windows Aero) and Windows 7 make use of the Desktop Compositing Engine (DCE) to implement new transparencies, live thumbnails, and various other on-screen effects. This is achieved by having each application window drawn off-screen and then composited. With the DCE enabled, Phindows' built-in double buffering is redundant. To save video RAM and increase performance, Phindows will detect if compositing is active. If compositing is active, double buffering will not be enabled by default.

In some cases under Windows Vista or Windows 7, the DCE will not be enabled, for example if the system does not have a DirectX 9.0 (Shader Model 2.0) capable video card, or if the user (or an application) has disabled it. In those cases, Phindows will do its own double buffering by default. For more information on Aero and the DCE, refer to the Windows Vista or Windows 7 documentation.

On Linux, you can run Phindows using WINE. See *www.winehq.org* for more information. In some cases, Direct3D may cause performance problems. If you encounter performance problems, you can set the Phindows double buffering method to `\"system memory\"` or `\"disabled\"`.

For context help on the different items in the Phindows Connect dialog, click on the **?** icon in the

title bar of the dialog and then click on the item you want more information on.

Phindows is supported on the following versions of Windows: Windows 2000, Windows XP, Windows Vista and Windows 7. Phindows runs well on the following additional platforms but not all features may be available: WINE under Linux, Windows 98 and Windows Me.

-F *cache_dir*

Specify the directory that `Phindows` can use for its image and font caches. Phindows creates two subdirectories called `cache` and `font`. If you don't specify a cache directory, the one shown in the Phindows Connect dialog is used. (See the note below.)

-h *height*

Initial window height (default: 480).

-H *t1*[,*t2*[,*t3*]]

Specify the mouse holdoff times (default: *t1*=1.2*9600/baud, *t2*=*t1*/2, *t3*=*t1*/4):

- *t1* — holdoff time for normal mouse motion.
- *t2* — holdoff time when the a mouse button is pressed.
- *t3* — holdoff time when a drag cursor is being moved.

-help

Phindows built-in help.

-I *icon_file*

The icon to use instead of the default icon.

-i *igrp*

Connect to a specific Photon input group (default: 1).

-K *key*

Specify a private key for data encryption. The key can be up to 31 characters. A matching key must exist in the `/etc/config/phkeys` file on the host machine you're connecting to, and the version of `phrelay` running on the host must support encryption. You can also specify a user (using the -U option) when you connect to a Photon service using the -s option. See the `phrelay` documentation in the

Neutrino *Utilities Reference* for more information about the `/etc/config/phkeys` file, and specifying keys and users.

- This option works on Phindows version 3.09 and greater, and `phrelay` running on QNX Neutrino 6.4.0 and greater. If you specify this option and attempt to connect to a version of `phrelay` that doesn't support encryption, the connection will fail. If `phrelay` does support encryption, but there's a key mismatch, Phindows displays Error: Permission problem on host.
- If `phrelay` is launched with the -e option (force encryption), then a private key must be specified on the Phindows command-line, or else the connection will fail.

-k

Start in kiosk (full-screen) mode. You can use `ctrl alt k` to toggle kiosk mode.

-M *size*

The maximum memory cache size, in kilobytes. If you don't specify a maximum memory cache size, the one shown in the Phindows Connect dialog is used. (See the note below.)

-m *commport*

Direct connect serial port (default: *com1*)

Setting this option turns on RLL compression and CRC error checking by default. You can turn it off with the -o option.

-N *number*

Set the number of messages buffered to allow write-ahead draws to `Phindows`. The actual number used is the lesser of this option and the -b option of `phrelay`. The default value is *20*. Use a lower value to save memory on the Neutrino host running phrelay at the expense of throughput, or, if memory is not an issue, a higher value to gain some additional throughput performance.

Adjusting this setting has the most effect when end-to-end response time is slow compared with throughput, such as over a modem or when there are many network hops between the local and remote ends.

-n

node_or_photon_resource Connect to an already-running Photon on the specified host, for example "`/dev/photon`". To get a list of available Photon sessions (if supported) use "List".

-o *options*

Options:

- 0 — no compression.
- 1 — BPE (Byte-Pair Encoding) compression.
- 2 — RLL (Run Length Limited) compression.
- 8 — Use CRC (Cyclic Redundancy Check) error checking.

Combine options by addition, e.g. to specify BPE and CRC, select *9*. You can't have BPE and RLL compression at the same time. If you select both, only BPE compression is used.

- If you don't specify this option, the RLL/BPE/CRC settings shown in the Phindows Connect dialog are used. (See the note below.)
- The command-line options -t and -m turn on RLL compression by default. The -m also turns CRC error checking on. To adjust these side-effects you can use the -o option afterwards. For example, to run with no compression when specifying a TCP/IP connection from the command line, you would type: *phindows -thost -o0*.

-O [Ala]

Disable (A) or enable (a) Alt+Tab pass through to the remote QNX system. By default, Alt+Tab key presses are only passed through to the remote system when Phindows is in kiosk (full-screen) mode. When this option is enabled, `Phindows` always passes the key presses through to the remote system (unless the Phindows window doesn't have focus). When the option is disabled, the Alt+Tab key press is processed by the local Windows system.

-O [Blb]

Disable (B) or enable (b) clipboard sharing. This option is enabled by default.

-O [Jlj]

Disable (J) or enable (j) antialiased fonts. This option is enabled by default.

Turning off antialiased font support can improve performance over a slow communications link, at the expense of visual accuracy. Antialiased text is rendered with 8 bits per pixel instead of 1, requiring more data transfer. Using RLL or BPE compression reduces the performance impact of displaying antialiased fonts.

-O [Olo]

Disable (O) or enable (o) offscreen context support. This option is enabled by default.

-P *palette_file*

Use a non-default color palette file (e.g. *%QNX_TARGET%*`\usr\photon\palette\grey.pal`).

-s *service*

Request that this Photon service be automatically started.

-T *size*

The maximum disk cache size, in megabytes. If you don't specify a maximum disk cache size, the one shown in the Phindows Connect dialog is used. (See the note below.)

-t *host_name* [:*port*]

TCP/IP address (and optional port) or host name of QNX host (e.g. 198.53.31.1, 198.53.31.1:4869, or myhost:4869).

Setting this option turns on RLL compression by default. You can turn it off with the -o option.

-U *userid*[:*password*]

Initially log into this QNX userid (used with -s).

-u

Unlocked mode (allows independent browsing of a Photon session). Phindows creates a separate input group when it

connects to an existing Photon session. This allows independent interaction with the Photon session.

-V[V...]

Increase the verbosity of Phindows status messages.

-v

Connect in view-only mode, which disables the keyboard and mouse.

-W "*Title*"

Window title (default: Phindows). The title may need to be quoted, and may not contain any spaces. Underscores will be converted to spaces. Special character sequences are supported for substitution of the remote host name (%H) and the remote photon session name (%S).

-w *width*

Initial window width (default: 640).

-X *position*

Initial x position of Phindows window (default: position from previous session).

-Y *position*

Initial y position of Phindows window (default: position from previous session).

-x *offset*

Create a Photon region at this x offset (default: 0).

-y *offset*

Create a Photon region at this y offset (default: 0).

-z *gzip_path*

Compress images using `gzip`. Images are compressed in their entirety, unlike RLL and BPE compression which are applied at the packet-level. Over a slow connection, this can improve performance.

Both `phrelay` and Phindows run the `gzip` utility using an external call. You must ensure that the `gzip` utility is available at both ends:

- On the Neutrino system, ensure that `gzip` is present in a directory listed in the *PATH* environment variable that is received by phrelay. In a typical self hosted Momentics installation, this is already the case.

- On the Windows system, the -z option passed to Phindows must specify the full path of the `gzip` executable and all arguments, for example:

  ```
  -z'c:\gzip.exe -d -f'
  ```

  or

  ```
  -z'"c:\Program Files\gzip\gzip.exe" -d -f'
  ```

  

  You should use single quotes to allow arguments to be passed, and double quotes around the executable name if it contains embedded spaces.

If you don't specify either -t or -m, Phindows displays the Connect dialog asking for communications parameters, and displays settings based on the last saved settings. When either -t or -m are specified, the Connect dialog isn't displayed, and settings are applied from the last saved session, or from defaults if there's no saved settings. Settings are saved to the `phindows.ini` file, whose location is shown in the About dialog.

Dittoing remote QNX Photon sessions

The normal mode of using Phindows (without the -n option) causes a private session of Photon to be started on the QNX machine you've connected to.

The normal mode of using Phindows (without the -n option) causes a private session of Photon to be started on the QNX machine you've connected to. Your Photon session is independent of everyone else's.

The -n command-line option lets a Phindows client see and interact with any Photon session already running on one of the QNX machines in the same QNX network you've connected to. An exact copy of the remote Photon user's screen is displayed on your Windows desktop. In addition, you can use your mouse and keyboard to interact directly with that remote Photon session. You can, in effect, take over the remote screen as if you were sitting there.

Remote support example

For example, if you want to provide support to a remote QNX site that's running Photon, dial up and log into that QNX machine using Phindows with a command line similar to:`phindows.exe -mcom2 -n/dev/photon`

Log in and start `phrelay` as follows:

On QNX 4:

```
/usr/photon/bin/phrelay
```

On QNX Neutrino:

```
/usr/bin/phrelay -x
```

At this point, `phrelay` is informed that a connection to an already-existing Photon session called `/dev/photon` is requested (i.e. the Photon running on the local console). The `phrelay` command creates a graphics region to overlay the entire console graphics region, so that both your mouse and the remote mouse can control the same cursor. You can then share that remote desktop with the remote user.

This type of remote connectivity is convenient for a variety of purposes, including:

- remote diagnostics
- remote monitoring
- remote technical support
- training.

Of course, there's no need to have a human actually sitting at the remote console. Nor is there a need even to have a physical screen and keyboard at the remote site. You can use Photon connectivity to control remote sites that are completely unstaffed. This not only saves on travel costs involved in remote diagnostics and maintenance,

but also saves in hardware costs at the remote site (since you can eliminate the cost of keyboards and display screens).

Connecting to a remote Photon session

Suppose you need to look at a Photon application running on some QNX machine other than the one with the modem you're connecting to. Not a problem. You can use the -n option to specify the full QNX pathname of any Photon session on that remote QNX network.

To interact with a Photon session on another node's console:

1. Connnect to the remote machine.
   - For a QNX 4 target, open a terminal and enter `phindows.exe -n//node_number/dev/photon`
   - For a QNX Neutrino target, enter `phindows.exe -n/net/node_name/dev/photon`
2. After connecting to the QNX network,(initially logged into the first node), start:
   - On QNX 4, `/usr/photon/bin/phrelay`
   - On QNX Neutrino, `/usr/bin/phrelay -x`

The `phrelay` program recognizes that you really want to be connected to a Photon on the second node, so it automatically migrates itself to that node, but continues to use the modem you dialed in on (i.e. the modem on the first node). Since QNX is an inherently distributed operating system, this happens seamlessly and efficiently.

Starting Photon sessions on other QNX nodes

Suppose the QNX machine you log into has lots of modems, but not a lot of spare CPU or memory. You want to start up a private Photon session and discover that node 2 has lots of spare resources (in QNX, any node will do).

To start a Photon session on another node:

On a QNX 4 target, in a terminal enter `phindows.exe -n//2/dev/ph+`

If you're connecting to a QNX machine that's using a color palette other than default.pal, you can specify the palette file using the -P command line option. The Windows display must be set to 256 colors (8-bit color) for this option to work.

For example, if the QNX machine you are connecting to uses the `grey.pal` palette file, use the following command-line option:

```
phindows.exe -P%QNX_HOST%\usr\photon\palette\grey.pal
```

Spanning a single Photon session across multiple screens

You can use both Phindows and `phditto` (which comes standard with Photon) to stretch a single Photon space across multiple desktops. Some (or all) of these desktops can be QNX computers, some (or all) of these desktops can be Windows desktops, and some (or all) can be X workstations as well.

The -x, -y, -h, and -w options are provided in both `phditto` and Phindows to make this possible. Here's how they work:

Suppose you have a QNX machine running Photon on a console in 1024x768 resolution. You now want to stretch the Photon space to include the 1024x768 Windows screen you've placed immediately to the right of the QNX screen (creating a 2048x768 Photon desktop). You simply need to start Phindows on the Windows machine with: `phindows.exe -x1024 -n/dev/photon`

When the connection is made (TCP/IP works well), you'll notice that the Photon desktop manager, which is normally on the bottom of the Photon screen, stretches to include the bottom of the right-hand MS-Windows screen as well. You can now take either mouse, start up new applications, drag them from screen to screen, and otherwise take advantage of your increased workspace.

With careful use of the -x and -y (and -h and -w) options, you can create many interesting combinations (an array of monitors forming one huge Photon display, multimedia presentations, etc.)

Sharing a Photon session (workgroup computing)

The normal mode of Dittoing someone else's Photon session (specifying -n) is to have a single cursor that can be controlled by either the local or remote user. Similarly, keystrokes from either keyboard can be entered into the application with keyboard focus, which is normally what you want for a remote training or debugging session.

But sometimes you may want the remote user to carry on working without necessarily being affected by what you're doing. You can do this by specifying the -u option on the Phindows command line. In this mode, your mouse, keyboard, and display are completely independent of the other user's console. Both of you see two cursors (your cursor is solid, while the other person's cursor is dimmed or ghosted), but the two cursors behave independently.

In Photon terminology, you're running in your own private *input group*. You're free to roam around the Photon space without affecting the other user. Your mouse clicks and keystrokes are directed to whatever application has input focus for your input group. You can therefore start up new applications anywhere in the Photon space (probably in a different Photon console to be polite). Unless the other user chooses to roam over to the part of the desktop you're working in, he or she is otherwise unaffected by your presence there.

There's no built-in limit to how many concurrent users can share the same workspace in this manner, although it's doubtful that you'd practically want to have more than two at any one time. It's far more likely that if multiple users are connected to a single QNX machine, they each run in their own private Photon workspace (the default behavior of Phindows) and communicate with each other using the built-in Photon connectivity facilities, such as Jump Gates and `msgpad` (QNX 4 only), and `phditto`. A single QNX machine can easily support dozens of such Phindows clients, provided it's fast enough and has enough RAM.

Enabling offscreen context support

The normal mode of Dittoing someone else's Photon session (specifying -n) is to have a single cursor that can be controlled by either the local or remote user. Similarly, keystrokes from either keyboard can be entered into the application with keyboard focus, which is normally what you want for a remote training or debugging session.

Offscreen context support is enabled by default, though you can turn it off with the -OO option.

Offscreen context support has some limitations:

- The support of the offscreen contexts via `phrelay` is limited to uncomplicated cases where applications do regular draws followed by periodic blits from offscreen to the main display. This means:
  - The `PdGetOffscreenContextPtr()` function isn't supported.
  - The creation of offscreen locks isn't implemented.
- These functions aren't supported by Phindows:
  - `PdGetOffscreenSurface()`
  - `PdDupOffscreenContext()`
  - `PdCreateDirectContext()`
- Using Phindows to connect to a Photon session where offscreen contexts currently exist or were previously orphaned will cause the software to attempt to re-create and re-draw those contexts. The accuracy that results depends on how well the applications currently running handle invalidation of their offscreen contexts.
- Offscreen context support is disabled when you connect to remote hosts running older versions of `phrelay`.

Using predefined Photon services

The -s command-line option simplifies the task of creating shortcuts to Photon applications within the Windows desktop.

By using the -s option, you can create an icon or shortcut on the Windows desktop that starts a Photon application automatically (within a private Phindows session). With proper specification of the remote Window manager options, it's possible to make that Photon application look like it's a native Windows application.

When `phrelay` runs on the QNX host machine, it looks up the Photon service specified with the -s option in a configuration file (`/etc/config/phrelay[.`*node*`]`). If a matching service is found, `phrelay` launches the specified Photon command instead of the default Photon desktop. You can specify optional Window Manager options, but the default mode is to start the remote Photon application such that it looks and behaves as if it were a native Windows application.

You typically use the -U option along with the -s option to specify a QNX userid to use when running the remote Photon command. If you don't give a userid, and the `phrelay` service doesn't specify a default one, Photon pops up a login dialog requesting the QNX userid before proceeding. By specifying a userid with the -U option, you can avoid this login dialog.

For example, if you create a Windows shortcut as follows:

```
phindows  -tx.x.x.x  -svpoker  -Ujoe
```

where the IP address *x.x.x.x* specifies the TCP/IP address of the Neutrino host, and the `phrelay` configuration file on the host (`/etc/config/phrelay`) contains the following line:

```
vpoker  %  /usr/photon/bin/vpoker
```

then *joe* could directly launch a Photon `vpoker` session (running as QNX userid *joe*) on his Windows desktop by simply clicking on the icon or shortcut.

Chapter 4
Configuring Phindows

Phindows uses environment variables to set basic configuration options.

This chapter describes how to configure Phindows Environment variables.

Configuring Phindows environment variables

The first time you use Phindows, you should set up the environment variables on your computer.

Phindows uses the following environment variables:

SYSTEMROOT
: The location of the system-wide phindows.ini file, if any. WINDIR is used if SYSTEMROOT is not set.

HOMEDRIVE
: Used in conjunction with HOMEPATH to determine the location of the user-specific phindows.ini file.

DOUBLE_BUFFER_DELAY
: A delay value in milliseconds. Double buffering will not blit to update the main window more often than this. The default is 5.

PHINDOWS_CLOSE_PROMPT
: If this variable is set, a simple yes/no confirmation prompt is always presented to the user when he or she closes the Phindows window. The default action (i.e. without this variable set) is to present a more complicated prompt only if a new remote Photon session was started during this run. The contents of this variable determine the confirmation prompt text. Related environment variables PHINDOWS_YES_TEXT and PHINDOWS_NO_TEXT can be used to customize the Yes/No button text.

Setting up an embedded target

In order to access an embedded target using Phindows, you must ensure that your embedded target is properly configured and that your buildfile contains at least the minimum set of required components. This topic provides an example of a basic buildfile.

Before you configure your target for remote access, it's a good idea to set up `photon` locally. This ensures that most of the necessary components that are required for remote access will be already present. If you run into problems, use the -D option of `phrelay` to troubleshoot your installation.

The following components are required in order to allow Phindows to connect to your embedded target:

In your `inetd.conf` file, add the following entry:

```
phrelay stream tcp nowait root /path_name/phrelay phrelay -x
```

where *path_name* is the correct path to `phrelay`. By default, `phrelay` is located in the following location: `/usr/bin/phrelay`.

In order to enable a private Phindows session, you must be able to log on. Ensure that you have the proper files installed::

- `/etc/group`
- `/etc/passwd`
- `/etc/shadow`
- `/etc/services`

You can verify that remote log in works by starting a telnet session with your target. If your `telnet` connection is successful, you can always remove `telnetd` and any related files.

The following listing shows a startup script that contains only the most basic components required to support a Phindows connection:

```
############################################################
##
## NOTES: A simple build script to allow a Phindows
##        connection on an embedded device.
##
############################################################

############################################################
## START OF BUILD SCRIPT
############################################################

inetd
phrelay
/etc/services
```

You must also include the `phlogin2` binary file that gets invoked by `phrelay` when a private Phindows session is created. The `phlogin2` binary executes a `ph` script which must be present in the following location `/usr/bin/ph`. Either create a local version of `phrelay` or create links to the following locations in your buildfile:

```
/usr/photon/bin/Photon
/usr/photon/bin/phlogin
#or phlogin2
```

The following files are required by `phlogin2`:

- `libAp.so.3`
- `libph.so.3`
- `libm.so.2`
- `libphexlib.so.3`
- `libimg.so.1`
- `libsocket.so.2`
- `libfont.so.1`
- `img_codec_png.so`
- `img_codec_jpg.so`
- `img_codec_gif.so`
- `/usr/share/faces/default`

These files should already be present if Photon is running on your target.

In the `ph` script, export the Photon environment variables and start the Photon components as you would from a startup script. Note that you should not start or restart `io-display` or `io-graphics`. Include the following script in your buildfile:

```
###########################################################
## START OF ph SCRIPT
###########################################################
export PATH=/proc/boot:/bin:/sbin:/usr/bin:/usr/sbin:/usr/photon/bin
export PHOTON_PATH=/usr/photon
export PHFONT=/dev/phfont
export HOME=/root
waitfor $PHOTON
if test ! $PHOTON -ef $PHOTON; then
print "Unable to start windowing kernel.\n"
exit 1
fi

pwm &
bkgdmgr &
sleep 1
shelf &
```

Make sure to include the following fonts, otherwise the login screen will not appear:

```
###########################################################
## font support
```

```
############################################################
/usr/photon/font_repository/pcs12.phf
/usr/photon/font_repository/pcterm14.phf
/usr/photon/font_repository/pcterm12.phf
/usr/photon/font_repository/phcursor.phf
/usr/photon/font_repository/tt0005m_.ttf
/usr/photon/font_repository/tt2003m_.ttf
/usr/photon/font_repository/tt2001m_.ttf
```

These are the basic requirements for connecting to an embedded device with Phindows.

Index