

NetGPS

A remote moving-map system for Oziexplorer.

1. Introduction & Document Information.....	3
1.1 Introduction.....	3
1.2 What can I use NetGPS for?.....	3
1.3 Changes.....	3
1.4 NetGPS on the Web.....	3
1.5 Copyright.....	3
2. Architecture.....	4
2.1 Overview.....	4
2.1.1. Description & Terminology.....	4
2.1.2. Notes.....	5
3. Setup.....	6
3.1 What you need.....	6
3.2 How to set up.....	6
3.3 Webserver Component.....	7
3.3.1. Installation Instructions.....	7
3.3.2. Testing.....	8
3.4 NetGPS Desktop.....	9
3.5 NetGPS PPC.....	9
4. Step-by-step; how to get started.....	10
4.1.1. Before you start.....	10
4.1.2. Remote: Desktop.....	10
4.1.3. Remote: PocketPC.....	10
4.1.4. Receiver.....	11
4.2 Testing.....	11
4.3 Setup Troubleshooting.....	11
5. Using NetGPS.....	12
5.1 NetGPS Desktop.....	12
5.1.1. General.....	12
5.1.2. Remote Mode.....	13
5.1.3. Receiver Mode.....	13
5.1.4. Saving Configuration.....	13
5.2 NetGPS PPC.....	14
5.3 Accounts, Usernames and Passwords.....	15
6. Troubleshooting.....	16
6.1 General.....	16
6.2 Known Issues.....	16
6.2.1. NetGPS Desktop.....	16
6.2.2. NetGPS PPC.....	16
6.2.3. NetGPS Webserver Component.....	17
7. Tested Equipment.....	18
7.1 General.....	18
7.2 Locations.....	18

7.3 Remote Configurations	18
8. About	19
8.1 Credits	19
8.2 Author	19
8.3 Status	19
8.4 The To-Do List	19
9. Appendix A: Remote Clients	20
10. Appendix B: Architecture Rationale	21
11. Appendix C: Webserver Component Technical Detail	22
11.1 Technical Description	22
11.1.1. Flowchart	22
11.1.2. Inputs and outputs	23
11.1.3. Notes	24
11.1.4. Security	24
11.1.5. Implementations	24
12. Appendix D: Change Log	25
12.1 Documentation	25
12.2 NetGPS Desktop	25
12.3 NetGPS Webserver Component	27
12.3.1. Perl Script	27
12.3.2. ASP.Net C# Page	27
12.3.3. Java Servlet	27
12.4 NetGPS Pocket PC	27
13. Appendix E: Data Transferred & Sample Costs	28
14. Appendix F: Perl and ASP.Net Setup	30
14.1 Perl	30
14.2 ASP.Net	30

1. Introduction & Document Information

1.1 Introduction

NetGPS is a companion system to Oziexplorer. It makes it possible to transmit coordinate data from a PC, across a network to another PC running Oziexplorer's moving-map system.

For example, you could use a laptop equipped with a mobile phone, dial up to the Internet, and then have another PC access your laptop and receive the moving-map info. Someone watching the receiving PC will see exactly where you are at all times, provided they have correctly set up Oziexplorer for moving-map capability. You need to be conversant with Oziexplorer and its moving-map operation before using NetGPS.

Note; NetGPS has not been written by the authors of Oziexplorer, and is in no way associated with them.

1.2 What can I use NetGPS for?

Seeing where people are on a digital map, as they drive around the world. Specific applications within that include realtime:

- spouse/offspring tracking;
- giving people directions;
- ?

Let your imagination run riot. I wrote this software because I thought it would be cool, not because I had a particular application in mind. If you think of one let me know.

Neither Oziexplorer nor NetGPS is suitable for J Average Homeuser who just wants to plug-n-play and not worry about COM 1 or 2, GPRS, datums, scanned maps or the difference between a filename and a directory.

1.3 Changes

See the Change Log, at the end of the document.

1.4 NetGPS on the Web

<http://www.gpsvehiclenavigation.com/GPS/netgps.php>

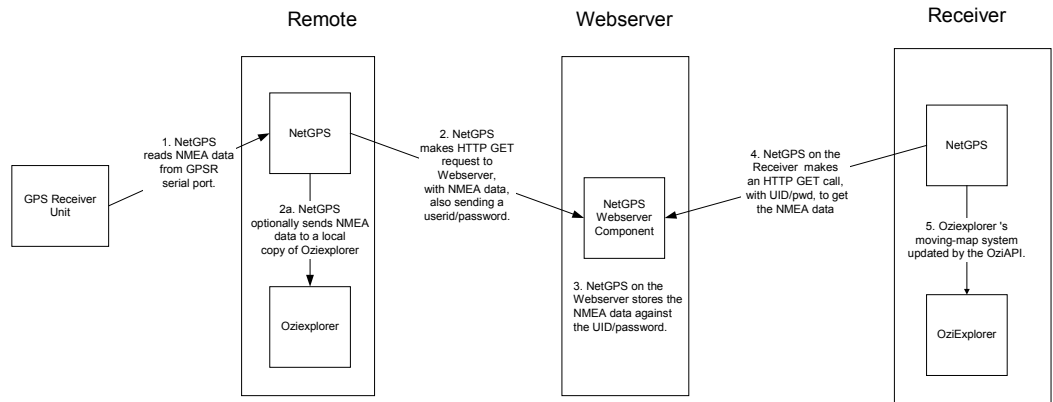
1.5 Copyright

Copyright Robert Pepper, all rights reserved.

2. Architecture

2.1 Overview

The basic architecture of the system is depicted below:



2.1.1. Description & Terminology

The **Remote** is a computer roving around the world, with a GPS receiver (GPSR), running a copy of NetGPS. NetGPS reads the position data (NMEA format) from the GPSR, and uses the HTTP protocol to send it and a username/password to a webservice which is running a NetGPS Webservice component. If a copy of Ozi is running on the remote, then NetGPS can send moving-map information to it, but that is not necessary for NetGPS operation.

The Remote can either be a laptop, or a PocketPC 2002 device.

The **Webservice Component** is a program run on a webservice that accepts the NMEA from the Remote and stores it against a username/password.

The **Receiver** also runs NetGPS, and must run a copy of Oziexplorer. See Section 3.1 for which version you need. The Receiver's NetGPS instance makes an HTTP request to the Webservice Component, and receives the NMEA data. It then sends this on to Oziexplorer, which uses it for a moving-map display.

The following tables summarise what NetGPS is:

<i>Remote</i>	<i>Webservice</i>
NetGPS Desktop	A Win32 program that acts as either Remote or Receiver.
NetGPS Webservice Component	Component that runs on webservice; implementations in Perl and ASP.Net supplied.
NetGPS PPC	A PocketPC 2002 program that can act as Remote, but not a Receiver.

“NetGPS” is used to refer to the entire system of Remote, Webservice Component and Receiver.

What does what:

<i>Remote</i>	<i>Webservice</i>	<i>Receiver</i>
NetGPS Desktop <i>or</i> NetGPS PPC	NetGPS Webservice Component	NetGPS Desktop

2.1.2. Notes

NetGPS Desktop may optionally use a web proxy server with Basic authentication. This is most useful for the Receiver, which may be forced to access the Web via a proxy server, for example most corporate networks.

NetGPS has been tested with a GSM dialup successfully. There was sufficient bandwidth to update the position fix every two seconds. You do not need a copy of Ozi on the Remote.

3. Setup

This section describes how to install and configure NetGPS.

3.1 What you need

<i>Item</i>	<i>Example</i>	<i>Notes</i>
<i>General</i>		
The NetGPS distribution.	N/A	http://www.gpsvehiclenavigation.com/ Supplied in a zipfile, named: netgps-x.x.x.zip where x.x.x is the version number. Contains NetGPS Desktop, PPC and Webserver Components. Everything!
<i>Remote</i>		
A GPSR that outputs NMEA data	Just about any Garmin, Magellan etc	-
A serial cable for that GPSR	-	To connect to your remote machine.
A remote machine (see Appendix A)	Laptop computer or Pocket PC 2002 device	Any Windows or PocketPC 2002 device will do.
<i>Network</i>		
A wireless network, and means of connecting both the Remote and Receiver to it.	The Internet, and GSM/GPRS, or 802.11.b are probably the best options.	It doesn't have to be a wireless network, but it does kind of defeat the purpose if it's not.
<i>Webserver Component</i>		
Webserver	Apache, IIS, PWS...	To execute the NetGPS Webserver Component. You could run a webserver on your desktop PC using a dialup line, it doesn't need to be a specialist hosted webserver.
NetGPS Webserver Component	Supplied in multiple implementations eg Perl, ASP.Net.	You can write your own in another language.
<i>Receiver</i>		
A Receiver machine.	Any Windows 9x/Me/2000/XP PC. NetGPS Desktop has only been tested on Windows 2000 SP2/3, but all the above should be fine.	Any PC capable of running Oziexplorer.
Oziexplorer 3.90.4.h2 or above, registered version.	-	From http://www.ozieplorer.com . Go pay the money.
Suitably calibrated moving maps.	AUSLIG 1:250k, if you're in Australia. You could also just scan a couple of pages of your local street directory in and use that, eg Melway.	Oziexplorer's help includes a how-to on map scanning and calibration.

3.2 How to set up

You'll need to set up your network connection, Oziexplorer and moving maps. If you aren't conversant with that, read Ozi's help and practice until you are. This guide assumes you understand basic Ozi moving-map operation, and is specifically about NetGPS.

Download the latest version of NetGPS from <http://www.gpsvehiclenavigation.com/GPS/netgps/> .

The NetGPS Distribution

Unzip `Netgps-X.X.X.zip`, where X.X.X is the version number, and you'll find the following directories:

```
\Desktop  
\Documentation  
\PocketPC  
\Webserver
```

If you do not see them, then try extracting the files by right-clicking the zipfile or by using Winzip's "Extract" toolbar button.

Setup Overview

1. Make sure the Webserver Component is installed somewhere and working.
2. Ensure you are familiar with Oziexplorer's moving-map capability. Test your Oziexplorer configuration by setting Ozi to read moving-map data from your GPSR before you start using NetGPS. Assistance with this step is outside the scope of this guide; RTFM.
3. Set up a Remote.
4. Set up a Receiver.

3.3 Webserver Component

3.3.1. Installation Instructions

Two webserver components are supplied; one is written in Perl and designed as a CGI program, and the other is written in ASP.Net using C#. You also can write your own.

Perl Script

It is assumed Perl is already installed on your webserver. If not, see Appendix F for assistance.

1. Decide on a directory that is outside the web root, and that the UID running perl has RWD access to.
2. Edit the Perl script, `netgps.pl` and change the variable defining the directory to the directory you defined in Step 1.
3. Test (see below).

ASP.Net Page

It is assumed that your webserver is already capable of running ASP.Net pages. If not, see Appendix F for assistance.

1. Copy `netgps.aspx`, `netgps.aspx.cs` to a directory of your choice. This must be in the web root and defined as an IIS Application.
2. Test (see below).

3.3.2. Testing

As all components do exactly the same thing, the test is the same. Change the URL to suit.

1. Run the steps in order.
2. Use a standard web browser.
3. `netgps.pl` is used as an example, substitute `netgps.aspx` or whatever as required.

Test Step 1

Test

Invalid Access

Input

`http://servername/netgps.pl`

Expected Result

GPSERROR:Invalid Parameters

Test Step 2

Test

No user data found for Receiver.

Input

`http://servername/netgps.pl?un=joe&pw=bloggs`

Expected Result

GPSERROR:No user data found.

Test Step 3

Test

Valid Remote input.

Input

`http://servername/netgps.pl?un=joe&pw=bloggs&cds=$GPRMC,065954,V,3244.2749,S,14809.9369,E,21.6,0.0,211202,11.8,E,S*07`

Expected Result

GPSOK

Test Step 4

Test

Valid Receiver input.

Input

`http://servername/netgps.pl?un=joe&pw=bloggs`

Expected Result

GPSOK\$GPRMC,065954,V,3244.2749,S,14809.9369,E,21.6,0.0,211202,11.8,E,S*07

Test Step 5

Test

Invalid credentials

Input

http://servername/netgps.pl?un=joe&pw=bloggs33

Expected Result

GPSError:Invalid credentials.

If the results aren't as expected, fix them!

3.4 NetGPS Desktop

Create a directory for NetGPS to live in, eg `c:\apps\netgps\`, and copy all of the files in the Remote&Receiver directory into that directory, ie:

```
netgps.exe  
netgps.ini
```

```
OziAPI.dll  
cswsk32.ocx  
mscomm32.ocx  
mscomctl.ocx  
tabctl32.ocx  
vbscript.dll  
msscript.ocx
```

Note; older versions of NetGPS used `mswinsck32.ocx` instead of `cswsk32.ocx` and did not require `OziAPI.dll`.

It is not possible to test NetGPS Desktop without a webserver component.

3.5 NetGPS PPC

1. Run `setup.exe` in the `\PocketPC\` directory.
2. Do not change the default installation path of `\program files\netgps\`

It is not possible to test NetGPS PPC without a webserver component.

4. Step-by-step; how to get started

4.1.1. Before you start

These steps assume that the webserver component is already installed.

What you need to know:

1. Webserver's IP address or hostname, and path to its Webserver Component.
NOTE; this is the path on the *webserver*, not your local machine! Added together, the "Server" and "Path" form a URL like <http://www.server.com.au/some/dir/netgps.pl>. Although netgps.pl is supplied, it is not run on either Remote or Receiver, only on the Webserver.
2. The userid/password you intend to use.

4.1.2. Remote: Desktop

1. **Connect the GPSR** to the Remote machine via a serial port.
2. **Check the GPSR is set to NMEA** communications, for example some Garmins may be set to communicate using Garmin's protocol.
3. **Load NetGPS Desktop on the Remote.**
4. **Select the "Remote"** radiobutton.
5. **Network Setup**; enter the webserver name, port and path to the Webserver Component, as well as the username and password.
6. **Check the TZ** is set to your timezone, eg X many hours +/- UTC (if Desktop)
7. Click **Start**.
The Remote will now be sending data to the webserver.

Optional; start Oziexplorer, and click the "Local Ozi" checkbox. NetGPS Dekstop will then send moving-map data to the Remote's Oziexplorer installation as well as over the network to the Webserver Component.

4.1.3. Remote: PocketPC

1. **Connect the GPSR** to the Remote machine via a serial port.
2. **Check the GPSR is set to NMEA** communications, for example some Garmins may be set to communicate using Garmin's protocol.
3. **Load NetGPS Pocket PC.**
4. **Network Setup**; enter the webserver name, port and path to netgps.pl, as well as the username and password.
5. Click **Start**.
6. The Remote will now be sending data to the webserver.
7. Optional; OziexplorerCE on the PPC, and click the "Local Ozi" checkbox. NetGPS PPC will then send moving-map data to the PPC's Oziexplorer installation. You will get a crosshairs marked "Vehicle 1" moving on your map, not the full moving-map functionality.

4.1.4. Receiver

1. **Start OziExplorer** on the Receiver machine. Do not start the moving map, but you may wish to show the Moving Map Control (from the “Moving Map” menu, select “Moving Map Control”.)
2. **Start NetGPS** on the Receiver.
3. **Select the “Receiver”** radio button.
4. **Network Setup**; enter the webserver name, port and path to Webserver Component, as well as the username and password.
5. **Check the TZ** is set to your timezone, eg X many hours +/- UTC.
6. **If you’re using a proxy server**, enter the details on the Proxy tab.
7. **Press Start.**

NetGPS will now retrieve the location data from the Webserver, where it has been placed by the Remote, and send it to Oziexplorer.

4.2 Testing

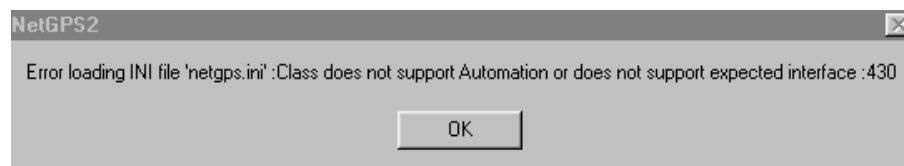
You don’t need to test NetGPS on a mobile device. You can run both Remote and Receiver on one PC, or two PCs over a network, provided one of them runs the Webserver component too. Some GPSRs can be put into simulation mode, and this works well for NetGPS; that’s how it was developed.

Time Delay

If you are familiar with moving-map systems you’ll know that the GPSR itself lags slightly behind the vehicle movement, and a moving-map has another lag on top of that. NetGPS introduces even more delay so the actual movement of the vehicle may be 2-7 seconds ahead of the moving-map display. You’ll get used to it.

4.3 Setup Troubleshooting

If you get this error message:



then there is a DLL conflict somewhere. To resolve, in order of preference:

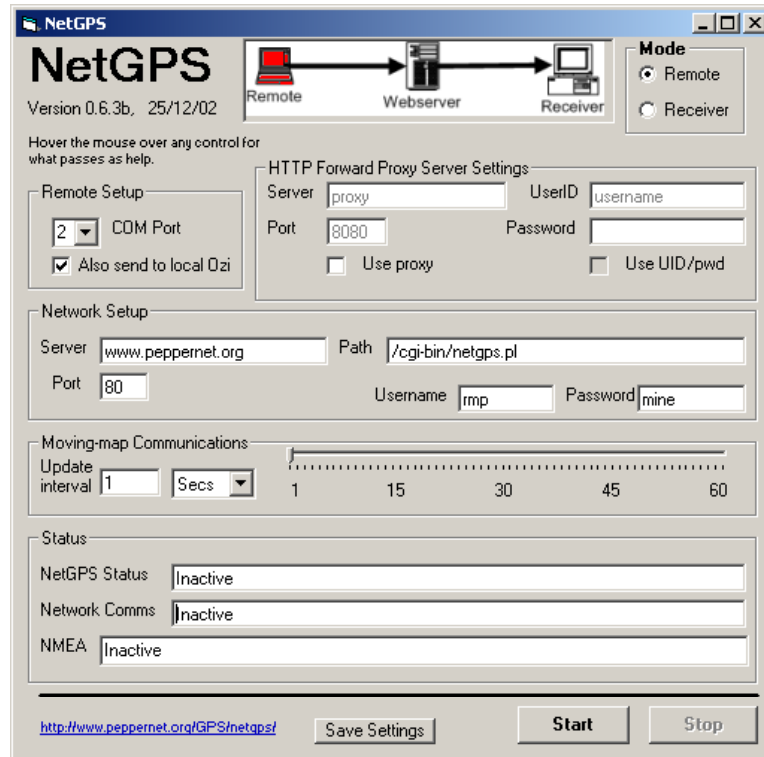
1. Search for the file vbscript.dll on your system. If the one supplied with NetGPS is newer, copy it over.
2. If you aren’t running MSIE 6, upgrade.
3. Install MDAC.

5. Using NetGPS

5.1 NetGPS Desktop

5.1.1. General

You can hover the mouse over any part of the screen and you may get a tip as to what the control does. This section is a bit more detailed. For reference, here is a screenshot:



Mode

Remote is for the machine with the GPS, travelling the world.

Receiver is the machine running Oziexplorer, getting data from the Remote.

Moving map update interval

How often the Receiver gets the GPS info from the Remote. Either use the slider, which goes from 1 to 60 units; the units being selectable as minutes or seconds. There's no point entering less than about 2 seconds, as the GPS itself only outputs data every 1.3 seconds or so.

Proxy Settings

If you wish NetGPS to use a proxy, then enter the details here and check the boxes. Authentication is basic only. The host can be an IP or hostname, anything that Windows itself is capable of resolving.

Port

The port the webserver listens on. By default, that's 80.

TZ

Your timezone, in terms of hours + or – UTC/GMT. For example “9.5”, “-6” and so on. Used to show the last update time, ie the last time the Remote updated the NMEA data, not the last time the Receiver retrieved it.

Status

NetGPS Status: general NetGPS status.

Network Comms: What the Remote and Receiver are up to.

GPS Output: The raw NMEA sentence, displayed on both Remote and Receiver. May be updated more frequently than the Update Interval.

The small coloured status rectangle is one of:

- Black; nothing happening.
- Green; it’s all good.
- Red; something is happening but it is not correct (tech note; the GPRMC A/V flag is set to A). Note that in simulation mode with Garmins this may be red. In that case, consider it green.

5.1.2. Remote Mode

Controls specific to Remote mode are:

COM Port

The serial port on the Remote that the GPS is attached to.

Also send to local copy of Ozi

The Remote can optionally send moving-map data to a local copy of Ozi, as well as make it available for the Receiver. If NetGPS is running, don’t try and get Ozi to access the GPS at the same time, use this option.

5.1.3. Receiver Mode

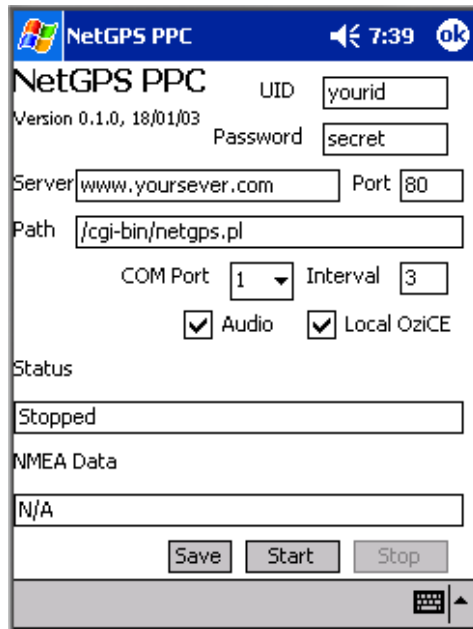
There are no controls specific to Receiver mode.

5.1.4. Saving Configuration

Use the “Save Settings” button to write all of NetGPS’ configuration, ie all of the input boxes to an ini file, netgps.ini. The exception is the proxy password. NetGPS will automatically save its settings when shut down anyway.

5.2 NetGPS PPC

There is no help for NetGPS PPC, just this manual.



NetGPS PPC

Version 0.1.0, 18/01/03

UID

Password

Server Port

Path

COM Port Interval

Audio Local OziCE

Status

NMEA Data

UID and Password

Credential used to access the Webserver Component.

Server

The server you are using.

Port

Port the Webserver Component is running on. 80 is the default.

COM Port

The serial port your GPS receiver is connected to.

Interval

How often, in seconds, you want NetGPS PPC to transmit your location. Maximum value of 64 seconds.

Audio

If you want voice error messages and/or a little beep if it goes wrong.

Local OziCE

Sends position data to OziCE. You get a crosshairs marked "Vehicle 1" which moves on the map, but not the full moving-map functions.

Status

A general indication of what's happening.

NMEA Data

The GPRMC NMEA sentence as read from your GPS receiver. Note that it is constantly read; the transmission interval may well be less frequent.

Save

Saves the configuration to netgps.ini. The settings are automatically saved anyway when you shut down.

5.3 Accounts, Usernames and Passwords

Accounts

The rules are:

1. If the Webserver component receives a username from the Remote that does not exist already, it creates an account for the user and assigns the password given.
2. The Receiver must supply an existing username/password. It cannot create a new one.
3. If, later, the Remote supplies the same username as before, but a different password, the Webserver component denies access.

The rules for usernames and passwords are:

Usernames

1. No more than 10 characters in length.
2. Only the characters a-z, A-Z and 0-9 permitted. No “-“.

Passwords

1. No more than 10 characters in length.
2. Only the characters a-z, A-Z, 0-9 and “-“ permitted.

6. Troubleshooting

6.1 General

1. The Remote and Receiver are entirely separate from each other. One can crash horribly, and not affect the other.
2. Problems with the webserver component will affect both but probably not cause a crash.
3. This is Windows. Therefore, there will be unexplained crashes, lockups and happenings. Follow the standard Windows error recovery procedure, to wit:
 1. Reset/kill the program(s) in question
 2. Curse
 3. Load everything up again
 4. Continue on your merry way.

6.2 Known Issues

6.2.1. NetGPS Desktop

As I haven't tested NetGPS all that well, I don't know what the error messages look like, but I'm sure they're horrendous and very uninformative. Nor have I trapped all the errors, so expect some to be still at large. Well, to be honest, expect *almost all* of them to be untrapped. Here are some things that are errors, bugs, or unfinished:

1. If Oziexplorer cannot proceed without the user answering a prompt, NetGPS is likely to freeze too. Check Ozi itself if NetGPS is frozen, for example Ozi may be prompting you for help with a map search.
2. Error trapping not complete. Expect instant death (not yours, NetGPS') if something untoward happens. Happily, the Remote doesn't care if the Receiver dies, or vice-versa. NetGPS may also freeze. If this happens, kill it and restart.
3. If NetGPS dies with an error, it may still be living as a orphaned process, condemned to evermore wander your system. Look for this by running Task Manager (CTRL-ALT-DEL in Windows NT/2000/XP) and then killing the NetGPS application from there. Windows 9x/Me users are best of rebooting.
4. If you have your GPSR set to a protocol other than NMEA, expect not very much to happen.
5. If Windows complains with weird error messages, plug them into your favourite search engine and see what happens. For example, if you get "System Error &H8007007E (-2147024770) The specified module could not be found." then try running `regsrv.exe msinet.ocx`.
6. I'm not convinced the status messages are always correct.

(it's amazing how honest you can be when you're writing free software!)

6.2.2. NetGPS PPC

1. Doesn't deal very well with poor responses from web servers.
2. Error status not entirely accurate.

6.2.3. NetGPS Webserver Component

Perl Script

None! Apart from the fact the supplied components are somewhat limited in function.

ASP.Net Script

Also none! And you know I'm being honest.

7. Tested Equipment

7.1 General

The equipment below has been tested with NetGPS and will probably work. There is almost certainly a lot more equipment out there that will work, but I can't test it all. If you'd like to send me some shiny new GPSRs or PCs then I will be happy to officially certify them as Most Probably All Right with NetGPS, but will need to keep them for a few years to make sure.

GPS Receivers	Garmin eMap at software rev 2.75 Garmin G12XL at software rev 4.58
Oziexplorer Versions	Desktop 3.90.4h2 (<i>do not use anything earlier than this version</i>) OziCE 1.10.9 (<i>do not use anything earlier than this version</i>)
Operating Systems	Windows 2000 with SP2 / SP3 Windows 98
Other	Works with NTL's "transparent" proxies.
PocketPCs	Compaq iPaq 3870 with Garmin eMap. O2 XDA.
Webservers	IIS 5 on Windows 2000 Apache (various versions)

7.2 Locations

Location Tested From (successfully!)

1. Dialup modem in Brisbane to GSM dialup in Melbourne
2. Corporate connection in Finland to GSM dialup in Melbourne
3. Cable modem connection in Sydney to GSM dialup in Melbourne.
4. Broadband connection from the UK to GPRS in Melbourne.
5. Dialup in Ottwa to GSM dialup in Melbourne.

All had useable updates. The Brisbane modem was the slowest, at about 4 seconds. The Sydney cable modem was about 1.5 seconds.

There haven't been any unsuccessful tests so far.

7.3 Remote Configurations

Tested so far:

1. IBM ThinkPad laptop, Win2K, Toshiba Bluetooth card connecting to a Nokia 6310i using Telstra's GPRS service.
2. IBM ThinkPad laptop, Win2K, Option GSM PC-Card, connecting to the Internet via a standard PPP dialup link over GSM.
3. IBM ThinkPad, Win2K and various 802.11b NICs.
4. Compaq iPaq, Bluetooth to Nokia 6310i, GPRS.

8. About

8.1 Credits

The main credit must go to Des Newman, for writing Oziexplorer. His release of the sample code finally prompted me to write NetGPS, which I had been meaning to get around to for about a year now.

Many thanks also to those who have assisted me with testing and bug reports, from all the way around the world and Australia, in particular Nigel Ward and Andrew Pepper.

NetGPS was written using the excellent, and free, SocketWrench TCP/IP libraries, available from <http://www.catalyst.com>. Highly recommended over the Microsoft Winsock libs.

8.2 Author

Robert Pepper <robert@peppernet.org>, a GPS user and 4WD enthusiast living in Melbourne, Australia. Please include the text "NetGPS" in the subject line of any emails you may wish to send.

8.3 Status

NetGPS is new code, and not well tested. If it doesn't work, crashes your computers, causes you frustration, scares your cat, puts SA back into GPS, sleeps with your partner, hurtles round the house screaming like a banshee or otherwise fails to perform as you expect, then so be it. Although I would like to know if it does any of the latter four activities, as I swear they aren't coded. The worst that should happen is that NetGPS itself dies.

8.4 The To-Do List

Part of the list includes:

- 1. A more comprehensive Webserver Component.**
 - Uses RDBMS, stores more than last position (history)**
- 2. Rewrite everything in .Net.**
- 3. Servlet Webserver component.**
4. Multiple vehicle view.
5. Web map view.
6. Have the Remote cache location data if it loses a fix.
7. Show Receiver whether the Remote has a GPS position fix or not.
8. Show heading, speed, alt on Remote.
9. Improve the status reports.
10. Better validation of user inputs.
11. Add a basic chat facility.
12. Add an installation system.
13. Better error handling.
14. Etc.

Items in bold are those I have any real intention of doing.

9. Appendix A: Remote Clients

The remote client must have:

1. A GPRS connected to the serial port
2. TCP/IP network connectivity to the Receiver.

The difficulty is in establishing a remote network connection that can be taken anywhere. A good choice for any meaningful distance is the Internet, but for shorter distances, wireless networks like 802.11b work well. Examples of possible configurations include:

- Laptop computer with 802.11a or 802.11b card.
- Laptop computer connected to the Internet by an IR link to a mobile phone with a modem (eg Nokia 7110, 8210, 8850)
- Laptop computer connected to the Internet by a Bluetooth link to a mobile phone with a modem (eg Ericsson T39, Nokia 8350, 6310, 6310i). This requires a Bluetooth card in the laptop.
- As above, but using GPRS instead of GSM. Not only is GPRS quicker, it's cheaper as you pay for the data transmitted, not the call time. But it is worth working out the cost of transmitting a megabyte, it is hardly cheap.

There are also PC-card modems that accept GSM SIM cards, eg the Option FirstFone (<http://www.option.com>).

10. Appendix B: Architecture Rationale

There are many ways to get the information from remote to receiver. The reasons the architecture that has been used are:

1. HTTP been used as the transmission protocol. It is reasonably lightweight. SOAP was considered but discarded as it is XML-based and therefore requires too much overhead. There was little point creating a brand new protocol, however simple, that would not be recognised by other systems.
2. A central webserver allows many Receiver to get the data of one Remote. Previous versions of NetGPS used a custom-coded webserver on the Remote, but this obviously causes an issue with the expensive and narrow bandwidth on the Remote.
3. Using a central server also means there is one fixed hostname to use, not a dynamic IP address as the Remote is likely to have. The Receiver may not be Internet-accessible, for example it may be behind a firewall. That is also another reason for using HTTP, and the reason for the HTTP proxy support.

11. Appendix C: Webserver Component Technical Detail

11.1 Technical Description

1. **Remote operation:** accept HTTP GETs with the following query string parameters:
 - a. cds – NMEA data (see below)
 - b. un – username
 - c. pw – passwordwhich will be from the Remote, as the Receiver does not pass the cds (coordinates, ie NMEA data) parameter. There is no need to URL-encode the NMEA data.
2. Store that data somehow. The Perl script does it by creating a file, with the filename derived from the user's name, and placing the password and the NMEA data in the file. The ASP.Net page does it by using the Application object. If successful, return the string:

GPSOK

as the page's body. Usual HTTP headers accepted.

3. **Receiver operation:** Accept HTTP GETs with the following query string parameters:
 - a. un – username
 - b. pw – passwordwhich will be from the Receiver. The component needs to find the username, and spit back the NMEA data as follows:

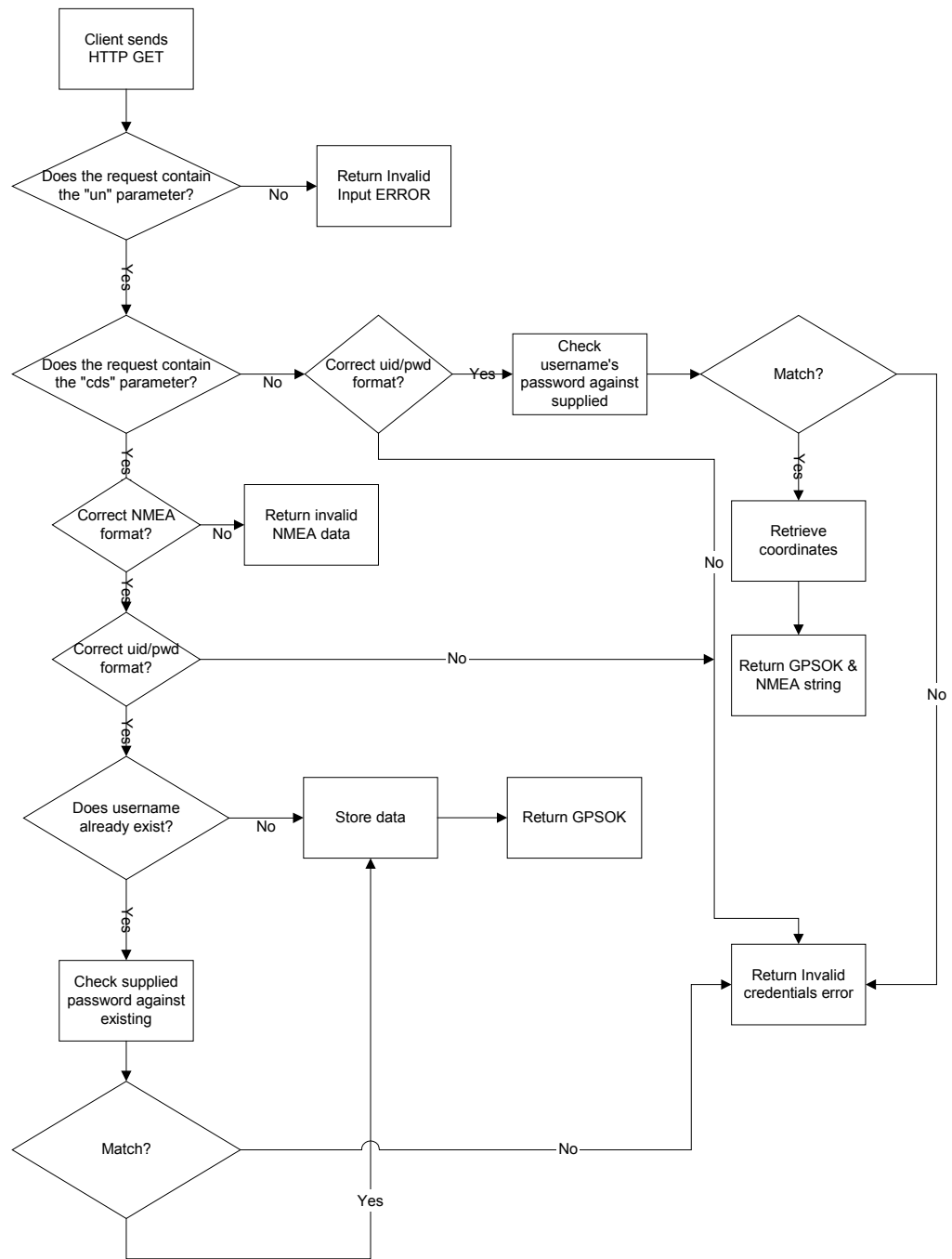
```
GPSOK$GPRMC,065954,V,3244.2749,S,14809.9369,E,21.6,0.0,211202,11.8,E,S*07
```

The text above is a real NMEA data string.

4. Although not strictly necessary, the supplied components do implement some security as follows:
 - a. If the username/password supplied by the Remote is not known, the data is stored.
 - b. If the username supplied by the Remote is known, the password supplied is compared against the one stored, and the data is only stored if the passwords matched.
 - c. If the username and password supplied by the Receiver do not match an existing username/password, an error is returned.

11.1.1. Flowchart

What the component has to do is shown by the following flowchart:



11.1.2. Inputs and outputs

Inputs

Just the standard querystring:

GET /yourpath&un=username&pw=password&cds=\$GPRMC.....

GET /yourpath&un=username&pw=password

For Remote and Receiver modes respectively.

Outputs

NetGPS expects the following outputs:

GPSOK	if in Remote mode
GPSOK\$GPRMC....	if in Receiver mode.
GPSError:	if there is an error.

in the body of the returned page. No HTML is used, eg <HTML>.

11.1.3. Notes

The Perl example persists the data to the filesystem. The ASP.Net example stores it in memory. Either works.

Cookies are **not** an option because they are only any good for maintaining state on a single client, and we're using two clients here, Remote and Receiver.

Disable caching if by default your system turns it on.

Disable any unnecessary headers, for example the ASP.Net page is set to not be part of Session State, so a cookie does not get set. There is no technical problem with this, but it does use up expensive bandwidth of the Remote.

11.1.4. Security

Other tasks the component should do, from a security perspective, are:

1. Validate the NMEA data. A regular expression is included in the Perl script.
2. Validate the username/password, and protect against hacking. For example, if any username is allowed, someone could create "hackme.sh", and then send faked NMEA data of something like "rm -rf *" which wouldn't be overly amusing. The Perl script guards against this by not placing files in the webroot, forcing prefixes and suffixes on the files, restricting the characters that a username/password can be comprised of, and running a regular expression over the NMEA data. Your components should do likewise, but your webserver's security is ultimately your own responsibility.

11.1.5. Implementations

Possible implementations for the component include:

- Servlets – servlets can store state between calls, so there would be no need to persist to the filesystem as per the Perl script.
- PHP;
- Anything that can write to a database.

12. Appendix D: Change Log

12.1 Documentation

06/07/03

- Fixed “cannot view speed” bug in NetGPS PPC.

04/07/03

- Added some hints on how to resolve DLL errors.

02/07/03

- Updated with PPC 0.2.0 changes.

21/01/03

- Started tracking changes

12.2 NetGPS Desktop

Version 0.7.0 04/06/03

- Added HTTP Accept and Accept-Language fields. Some weird webserver required it.

Version 0.6.9 23/01/03

- Fixed bug with crashing when the NMEA string didn't have a lock.

Version 0.6.8 21/01/03

- Changed name to “NetGPS Desktop”

Version 0.6.7 28/12/02 (Code and Doc)

- Fixed error in supplied INI file
- Made the TZ box a little wider so it fits +9.30 etc in properly.
- Clarified how to unzip the distribution file.

Version 0.6.6 28/12/02 (Code and Doc)

- Fixed potential error where the update interval was cleared and the system tried to process data, resulting in an error. Now if this happens the interval is set to 1 until the user updates it again.
- Added timezone tooltip (hover-help)
- Changed timezone 5.5 to 5.30 (typo)
- Improved tab order.

Version 0.6.5 28/12/02 (Doc)

- Updated list of files in 3.2.1 to include the two new libs as per 0.6.4.
- Added extra credits.

Version 0.6.5 28/12/02 (Code and Doc)

- Fixed bug with timezones.
- Changed timezone input to combo box.

Version 0.6.4 27/12/02 (Distribution)

- Added missing DLL/OCXs so the error about the INI file doesn't appear on startup.

Version 0.6.4 26/12/02 (Code and Doc)

- Added timestamp of last update by Remote, including timezone.
- Improved handling of invalid NMEA data when GPSR is attempting to get a fix.
- Added a colour-coded status block.

Version 0.6.3b 25/12/02 (Doc)

- Added example costs.

Version 0.6.3b 25/12/02 (Code and Doc)

- Fixed a bug with the proxy server facility.

Version 0.6.3a 24/12/02 (Code and Doc)

- Changed NetGPS to use HTTP 1.1 instead of 1.0. It's not very much more in terms of overhead anyway and allows compatibility with host-header HTTP 1.1 servers. HTTP 1.0 is still used via proxies.

Version 0.6.3 24/12/02 (Code and Doc)

- Improved error message if the server returns what NetGPS deems to be an invalid response.
- Clarified what the Path means; it's not the local path to netgps.pl, it's part of the URL.

Version 0.6.2 24/12/02 (Code and Doc)

- Added a "Save settings" button. Still saves on exit though.
- Fixed bug with send to local Ozi. It now works if you stop/start.
- Saves NetGPS password to NetGPS.ini.

Version 0.6.1 23/12/02 (Code and Doc)

- Documentation revision, error corrections, added headings.
- Made hyperlink on NetGPS interface clickable.

Version 0.6.0 21/12/2002 (Code and Doc)

- Added a central Webserver component, removed webserver on Remote.
- Uses new OziAPI, requiring Oziexplorer 3.90.4h2
- Redesigned user interface
- Added option for HTTP proxy use, and authentication
- Fixed lots of bugs
- Probably introduced new ones

Version 0.6.0 22/05/03 (Doc)

- Updated doc with new URL
- Added references to help people get started with Perl and ASP.Net.

Previous versions used a webserver on the Remote.

12.3 NetGPS Webserver Component

12.3.1. Perl Script

0.1.0

- Initial release.

12.3.2. ASP.Net C# Page

0.1.1

- Added Response.BufferOutput = True
- Removed label property change, substituted Response.Write as the label added some useless extra HTML.
- Fixed a bug with input parsing if just a username and no password was supplied.
- Fixed bug with application object not updating.

0.1.0

- Added ASP.Net webserver component

12.3.3. Java Servlet

In development.

12.4 NetGPS Pocket PC

0.2.1 06/07/03

- Fixed non-display of speed from 0.2.1 when using Ozi CE local.

0.2.0 02/06/03

- Added extra COM port options (total of 9) to permit operation with for example the Bluetooth GPSRs that operate on COM5.
- Now using the new OziCE API that permits proper moving-map operation; previously used vehicle display.

0.1.1 23/01/03

- Fixed bug with CDbI exception when the NMEA data was invalid and OziCE local was checked.

0.1.0 21/01/03

- Initial release

13. Appendix E: Data Transferred & Sample Costs

Wireless Internet access is not cheap, and 802.11b networks aren't the answer as coverage is spotty and virtually non-existent out of urban areas.

To help you calculate NetGPS' network usage the following information is supplied. Please note it is all *approximate* and your particular conditions will dictate your usage and thus costs.

This is a sample NetGPS Remote request:

```
GET /cgi-bin/netgps.pl?un=jbloggs&pw=secret&cds=$GPRMC,143238,V,3731.0466,S,14523.8828,E,37.8,191.0,201202,11.8,E,S*0B HTTP/1.1
User-Agent: NetGPS
Host: shogun:80
```

That is approximately 170 bytes of data. The return string is about 103 bytes, mostly the webserver's HTTP headers. Making some more assumptions, here are the sample costs for GPRS usage, which is billed by data transferred not by time.

Approximate GPRS Costs for Remote

Approximate size of NetGPS request	170	bytes		
Approximate size of NetGPS response	130	bytes		
Total	300			
Cost per kb	\$0.02			
Per-second updates			Duration	
			1 Minute	1 Hour
			5 Hours	
Updates per period	60	3600	18000	
Data transferred	18000	1080000	5400000	b
	17.6	1054.7	5273.4	kb
	0.017	1.030	5.150	Mb
Cost	\$0.35	\$21.09	\$105.47	
10-second updates				
Updates per period	6	360	1800	
Data transferred	1800	108000	540000	b
	1.8	105.5	527.3	kb
	0.002	0.103	0.515	Mb
Cost	\$0.04	\$2.11	\$10.55	

All figures in this chart are approximate only and should be used as a guide.

Notes

1. Even though you set it to one-second updates, NetGPS Remote may not be able to transfer data that quickly.
2. If you have other programs running, for example an instant messaging client, email client etc then obviously they will use bandwidth too.
3. This excludes Receiver costs, as these are not likely to be over a wireless (expensive) link.
4. Normal GSM phones are billed for time regardless of the data transferred.
5. Note that your cellular service provider may bill data call at a premium rate.
6. Note that your GPRS provide may charge a flagfall.
7. Note that you should figure out your own costs!

14. Appendix F: Perl and ASP.Net Setup

These instructions assume your webserver is already set up to run either Perl scripts or ASP.Net pages. If not, your friendly webmaster can assist. If you are that webmaster and the world of webmastering is new to you, refer to these links to help. It's not hard, and it's a useful technical skill to learn.

14.1 Perl

The steps are:

1. Install Perl
2. Ensure it is configured to run CGI scripts. The installer will probably do this automagically.

Perl for Windows can be found at <http://www.activestate.com>, and the generic Perl site is <http://www.perl.com>.

A useful FAQ is:

<http://aspn.activestate.com/ASPN/Reference/Products/ActivePerl/faq/Windows/ActivePerl-Winfaq6.html>

The supplied script will run on any operating system capable of running Perl, which is pretty much any you can think of!

14.2 ASP.Net

This script runs only on Windows boxes capable of running ASP.Net, which essentially means IIS or PWS systems. To enable your standard IIS/PWS installation for running ASP.Net, and in fact any other .Net program, you need the .Net Framework, available from:

<http://msdn.microsoft.com/netframework/>

Install that, and go. Useful ASP.Net support sites include <http://www.15seconds.com>.