Access Point and Access Server

User's and Developer's Guide

Bluegiga Technologies

Access Point and Access Server: User's and Developer's Guide
by Bluegiga Technologies

Published 2009-04-17 (4.0)

Table of Contents

List of Tables

Chapter 1. Introduction

Bluegiga Access Server product family offers cutting-edge wireless Bluetooth routers, Access Points and management tools - enabling you to create efficient and scalable networks. The open and adaptable platform enables you to meet your applications' and customers' needs.

Bluegiga Access Point 3201 is a size-optimized access device targeted at business applications. The product is designed to fit into wireless Bluetooth applications where network performance, reliability, scalability and easy management are important design drivers.

Access Point 3201 is an evolution from Bluegiga's extremely reliable and successful Access Server product family. Access Point product software and user interface make it compatible with Bluegiga Access Servers. Access Points can be remotely managed from a centralized location with Bluegiga Solution Manager (BSM), a web-based remote management and monitoring platform.

Access Server is a cutting edge wireless Bluetooth router. It supports multiple communication standards including ethernet, WiFi, and GSM/GPRS enabling full media-independent TCP/IP connectivity. Access Server is easy to deploy and manage in existing wired and wireless networks without compromising speed or security. For rapid deployment, Access Server configurations can easily be copied from one device to another by using USB memory dongles. The device can be fully managed and upgraded remotely over SSH secured links. Large numbers of Access Servers can easily be controlled using Bluegiga Solution Manager (BSM).

Usage scenarios and applications:

- Medical and health device gateways
- Bluetooth proximity marketing
- Point-of-sale and retail systems
- Telemetry and machine-to-machine systems
- Industrial Bluetooth gateways

Key features:

- Open Linux platform for adding local customer applications
- Turn-key applications for Bluetooth networking and Bluetooth proximity marketing
- Supported Bluetooth profiles: SPP, ObjP, FTP, PAN, LAP, DI
- Software-configurable range to up to 100 meters
- External and internal antenna options
- Supports all key communication medias:
  - Bluetooth
  - Ethernet
  - WiFi, GSM and GPRS supported via USB (Access Server supports also via Compact Flash)
  - USB and RS232
- Fast and easy to install

- Uncompromised security: SSH, firewall, and 128 bit Bluetooth encryption
- Bluetooth, CE, FCC and IC certified
- Compliant with Bluetooth 1.1, 1.2 and 2.0 Specification

1.1. Licenses and Warranty

Warning

Bluegiga Technologies is hereby willing to license the enclosed WRAP product and its documentation under the condition that the terms and conditions described in the License Agreement are understood and accepted. The License Agreement is supplied within every WRAP product in hard copy. It is also available on-line at http://www.bluegiga.com/terms_and_conditions. The use of the WRAP product will indicate your assent to the terms. If you do not agree to these terms, Bluegiga Technologies will not license the software and documentation to you, in which event you should return this complete package with all original materials, equipment, and media.

Some software components are licensed under the terms and conditions of an open source license. Details can be found from http://gpl.bluegiga.com/ or in directory `/doc/license/` in SW CD-ROM or SDK DVD-ROM.

The Bluegiga WRAP Product Limited Warranty Statement is available on-line at http://www.bluegiga.com/terms_and_conditions.

1.2. Bluegiga Technologies Contact Information

Please see http://www.bluegiga.com/ for news and latest product offers. For more information, contact <`sales@bluegiga.com`>.

Please see http://techforum.bluegiga.com/ for software and documentation updates.

Please contact <`support@bluegiga.com`> if you need more technical support. To speed up the processing of your support request, please include as detailed information on your product and your problem situation as possible.

Please begin your email with the following details:

- Access Server product type
- Access Server product serial number
- Access Server software version
- End customer name
- Date of purchase

  Note: You can generate the product information by navigating to WWW Setup ⟶ Advanced ⟶ System Information ⟶ Collect info for support request.

Chapter 2. Getting Started

Access Point and Access Server can be controlled in four ways:

- by using Bluegiga Solution Manager (see BSM documentation for details)
- by using the WWW interface
- by entering commands and using applications at the shell prompt
- by sending and/or retrieving files to/from the device.

Note: The default username is **`root`** and the default password is **`buffy`**.

2.1. Powering Up

To get started with Access Point or Access Server, connect it to your local area network (LAN) by using an ethernet cable, and connect the power adapter. The unit will power up and retrieve the network settings from your network's DHCP server.

Access Point and Access Server will also use Zeroconf (also known as Zero Configuration Networking or Automatic Private IP Addressing) to get a unique IP address in the 169.254.x.x network. Most operating systems also support this. In other words, you can connect your controlling laptop with a cross-over ethernet cable to Access Server, then power up Access Server, and the devices will automatically have unique IP addresses in the 169.254.x.x network. With Access Point, also a direct ethernet cable works.

Note: If you need to configure the network settings manually and cannot connect Access Server first by using Zeroconf, you can do it by using the management console. For more information, see Section 2.3.1. Access Point, however, does not provide user access to the management console. You can configure static network settings by sending the settings in a management packet for example using a USB memory dongle. See Section 3.9.4 for more information.

The physical interface locations of Access Point are described in Figure 2-1.

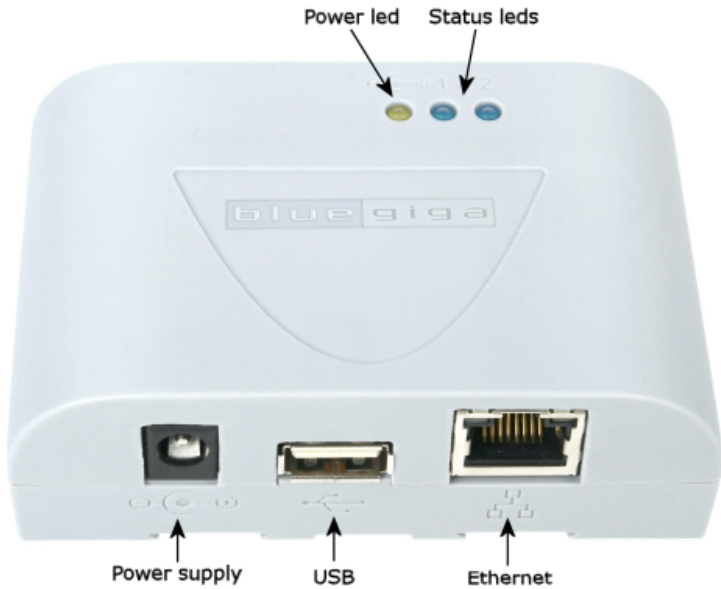

Figure 2-1. Access Point Interfaces

Note: There is no power switch in Access Point. The adapter is the disconnection device; the socket-outlet shall be installed near the equipment and shall be easily accessible. Unplug and plug the power adapter to switch the power on and off. The power led in Figure 2-1 is on when the power adapter is connected.

The physical interface locations of Access Server are described in Figure 2-2 and Figure 2-3.

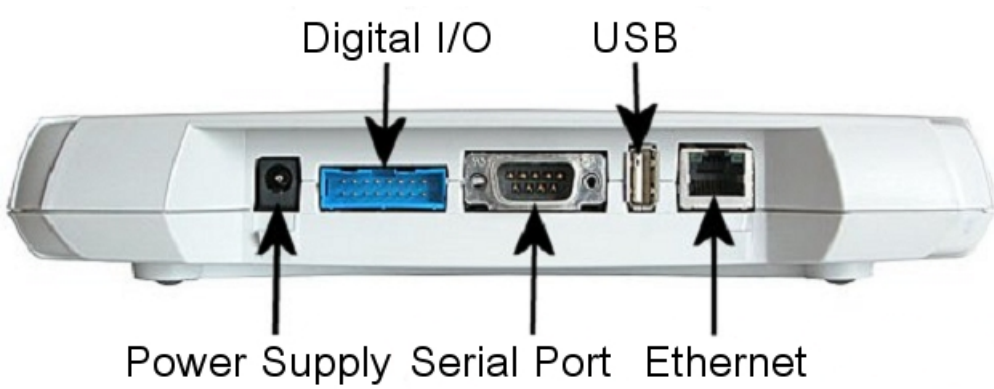

Figure 2-2. Access Server Connectors

Note: There is no power switch in Access Server. The adapter is the disconnection device; the socket-outlet shall be installed near the equipment and shall be easily accessible. Unplug and plug the power adapter to switch the power on and off. The power led in Figure 2-3 is on when the power adapter is connected.

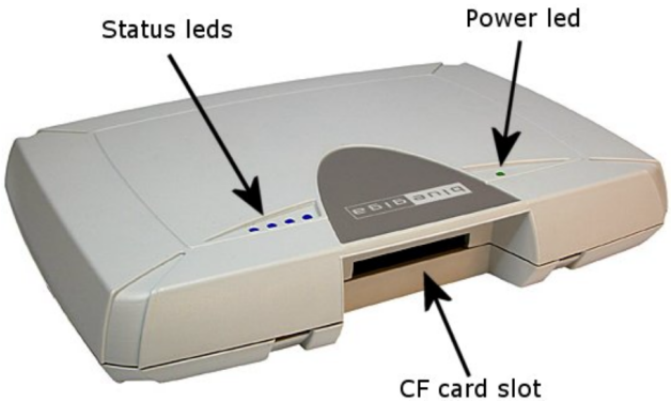

Figure 2-3. Access Server leds

All the blue status leds are turned off and the rightmost blue led (closest to the power led, led number 1 in Access Point) blinks on four second intervals when the boot procedure is finished and the unit is ready to be connected. Bluetooth led, (led number 2 in Access Point and blue led furthest away from power led in Access Server) blinks quickly every 30 seconds indicating Bluetooth service activity.

2.2. WWW Interface

Most Access Point and Access Server functionality can be controlled through the WWW interface by using any standard WWW browser.

The wrapfinder application (see Figure 2-4), available for the Windows operating system from Bluegiga Techforum (http://techforum.bluegiga.com/) provides an easy-to-use interface for finding Access Points and Access Servers (with SW version 2.1.0 or later) in the local area network.

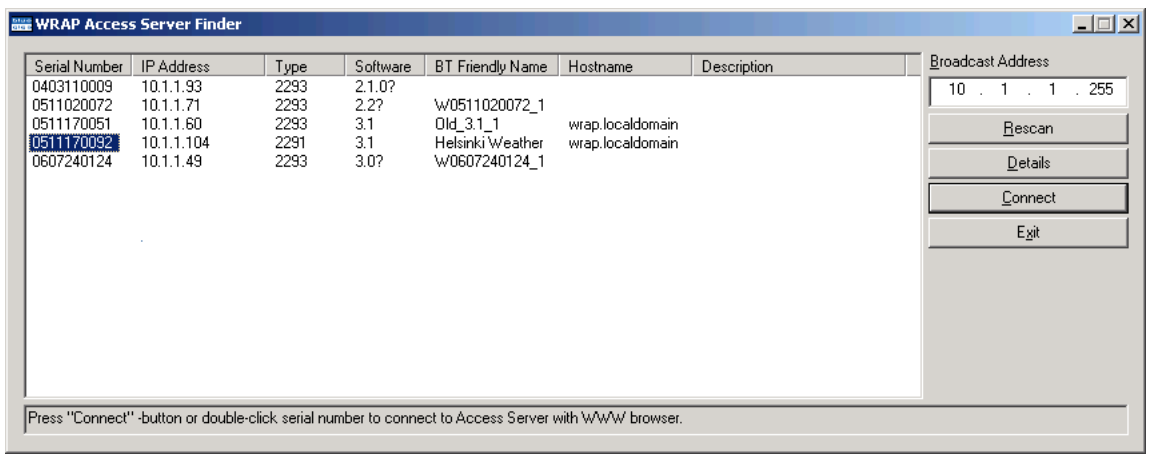

Figure 2-4. Access Server Finder Application

When wrapfinder is launched, it automatically identifies the broadcast address of the network it runs in and sends a special query packet (UDP broadcast) to units. The most important information in their responses is then shown in table format.

You can change the broadcast address used for finding Access Points and Access Servers. A new scan can be done by clicking Rescan.

Select a unit by clicking its serial number, and click Details to see more information (such as all Bluetooth addresses and friendly names). See Figure 2-5 for details.

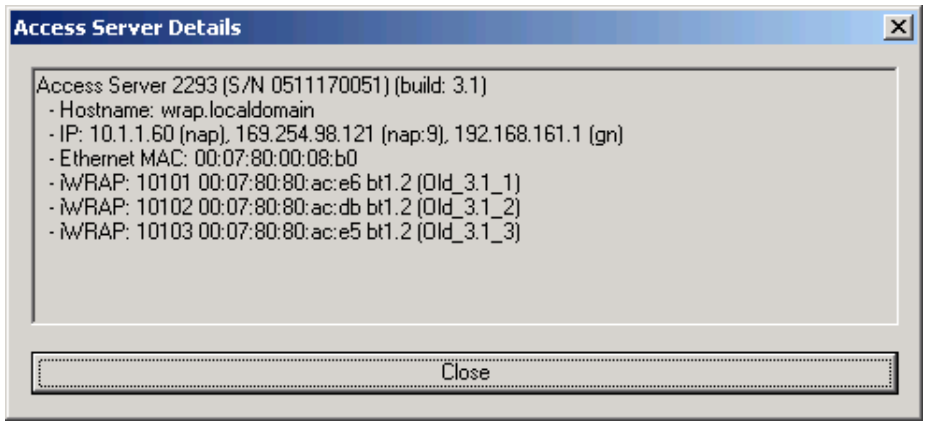

Figure 2-5. Details Dialog of Access Server Finder

Click Connect or double-click a serial number to connect to the selected Access Point or Access Server by using a WWW browser.

Click Exit to close the program.

Note: To find Access Server's IP address without wrapfinder, see Section 2.3.2.

To access the WWW interface, enter the IP address of Access Point or Access Server to the browser's address field and press **Enter** (see Figure 2-6).

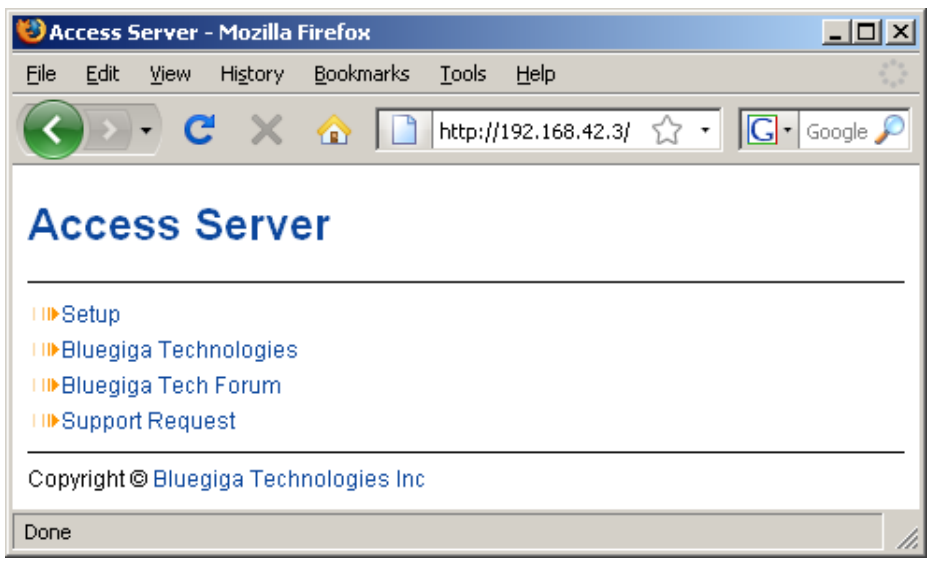

Figure 2-6. WWW Interface

From the top-level page, click Setup to log in to the configuration interface. The default username is **root** and the default password is **buffy** (see Figure 2-7).

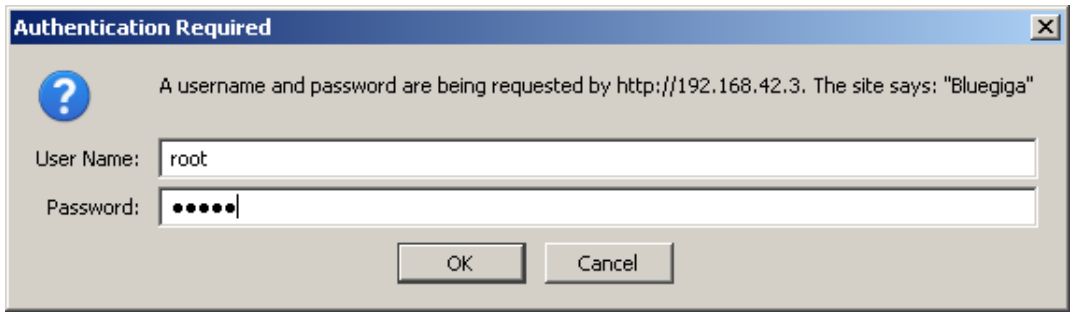

Figure 2-7. WWW Login Prompt for Access Server Setup

After logging in, you can configure several settings (see Figure 2-8). These are discussed in detail in Section 2.4.

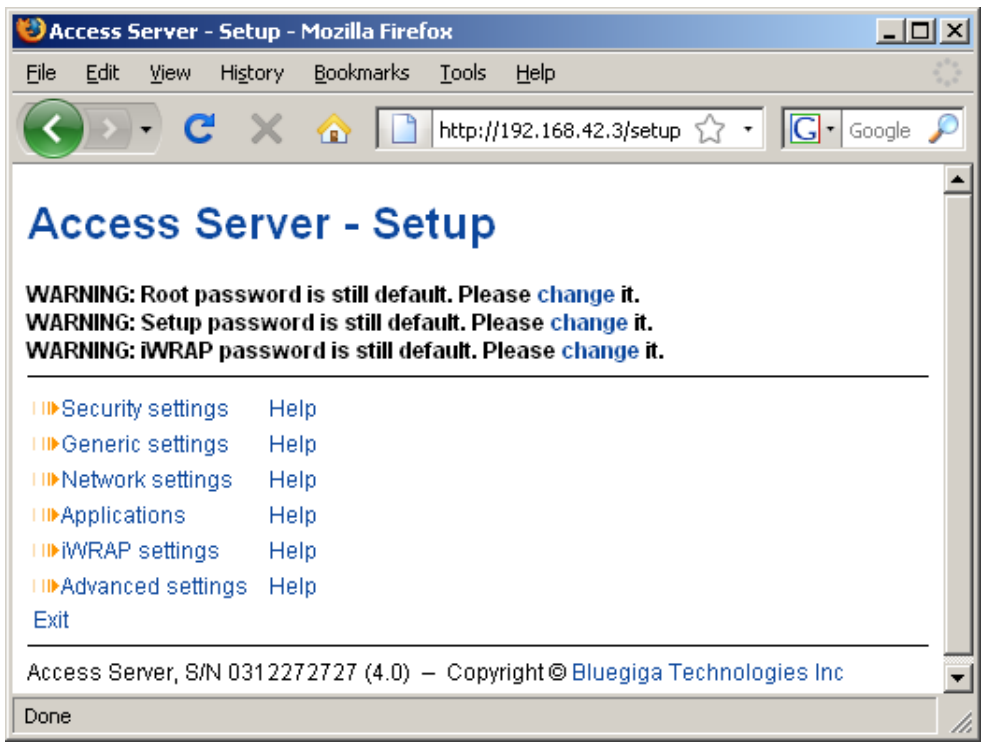

Figure 2-8. The WWW Configuration Interface

2.3. Shell Prompt Access

Shell prompt access may be needed for advanced controlling operations that cannot be performed by using the WWW interface. You can get to the shell prompt by using SSH. When you are connected to the same LAN network with your Access Server or Access Point, you can find its IP address using the wrapfinder application (see Section 2.2 for details).

You can use SSH to get shell prompt access also using Bluetooth LAN Access or PAN profile. Access Server and Access Point can be seen in Bluetooth inquiries as "Wserialno_n", where "serialno" is the serial number of the device and "n" is the number of the Bluetooth baseband in question (model 2293 has three Bluetooth basebands, any of which can be connected). After you have connected to the server (no PIN code, username or password is needed), establish an SSH connection to the device at the other end of the connection. You can use the wrapfinder application to find the IP address (see Section 2.2 for details).

Note: Bluetooth LAN Access and PAN profiles are disabled by default. Use the WWW interface to enable them, if needed. The PAN profile can also be enabled by sending the `enable-pan.noarch.wpk` file (available on-line at http://bluegiga.com/as/current/enable-pan.noarch.wpk) to Access Server by using Bluetooth Object Push profile or by inserting a USB memory dongle with the file in its root directory to Access Server's or Access Point's USB port.
Note: The default username is **`root`** and the default password is **`buffy`**.

2.3.1. Management Console (Access Server only)

If you do not have a Bluetooth LAN/PAN client and if Access Server is not connected to your LAN, or if you do not know the IP address given to Access Server, you can get the first shell prompt access by using the management console. The management console is only needed to change the network configuration settings if you cannot configure the network by using DHCP or Zeroconf. The management console is connected to Access Server with a serial cable. After you have configured the network settings by using the management console, all further controlling activities can be performed remotely using SSH sessions over ethernet or Bluetooth LAN/PAN connection.

To setup the management console, proceed as follows:

1. Have a PC with a free COM port.
2. Power off Access Server.
3. Configure your terminal application, such as HyperTerminal in Windows, to use the settings below for your computer's free COM port

Setting	Value
Speed	115200bps
Data Bits	8
Parity	None
Stop Bits	1
Flow Control	None

Table 2-1. The Management Console Port Settings

4. Connect the serial cable shipped with Access Server to your PC's free COM port.
5. Connect the serial cable to the management (user) port in Access Server (see Figure 2-2).
6. Power on Access Server.
7. Enter letter **b** in the terminal application during the first five seconds.
8. The management console is now activated and you can see the boot log in your terminal window.

Note: The boot process may stop at the following U-Boot prompt:

```
Hit any key to stop autoboot:  0
U-Boot>
```

If this happens, enter command **`boot`** to continue to boot Linux.

9. Wait for the device to boot up and end with the following prompt:

```
Please press Enter to activate this console.
```

10. Press **Enter** to activate the console. You will be logged in as **`root`** in directory `/root`:

```
[root@wrap root]
```

11. You can now control Access Server from the management console.

2.3.2. Accessing Remotely

When Access Server or Access Point is connected to a LAN, it tries to get the IP address by using DHCP and Zeroconf by default. You can then use the wrapfinder application to find the IP address (see Section 2.2).

With Access Server but not Access Point, if you cannot get the IP address by using the wrapfinder, another way to see the IP address is to connect with a management console (see the previous section), power on the unit and, after the system is up and running, give the **ifconfig nap** command. The `inet addr` field for the `nap` interface contains the IP address of Access Server. For example, in the following capture from the management console, the IP address is `192.168.42.3`.

```
[root@wrap /]$ ifconfig nap
nap       Link encap:Ethernet  HWaddr 00:07:80:00:BF:01
          inet addr:192.168.42.3  Bcast:192.168.42.255  Mask:255.255.255.0
          inet6 addr: fe80::207:80ff:fe00:bf01/64 Scope:Link
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:12635 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1686246 (1.6 MiB)  TX bytes:1640 (1.6 KiB)
          Interrupt:24 Base address:0xc000
```

You can use this address to connect to Access Server remotely over SSH, SCP or SFTP.

Note: The default username is **`root`** and the default password is **`buffy`**.

2.3.3. Transferring Files to/from Access Server

You can transfer files to and from Access Server and Access Point by using, for example:

- SCP (secure copy over SSH).
- SFTP (secure FTP connection over SSH).
- FTP (plain FTP connection).

Note: FTP is disabled by default for security reasons. Use SFTP instead. FTP server is not installed by default. You can install it from software package `ftpd`. See Section 3.2 for information about installing software components.
Tip: If enabled, use the integrated FTP client on the Internet Explorer (type ftp://root:buffy@wrap-ip-address/ in the address bar).

- Bluetooth OBEX (Object Push and File Transfer Profiles) to/from directory `/tmp/obex` in Access Server or Access Point.
- NFS (mount an NFS share from a remote computer as a part of Access Server's or Access Point's file system).
- SSHFS (mount an Access Server or Access Point directory over SSH as a part of any other Linux host file system).

  To download and install SSHFS, visit http://fuse.sourceforge.net/sshfs.html.
- CIFS (mount a Common Internet File System share from a remote computer as a part of Access Server's or Access Point's file system). A CIFS client, available in a separate software packet `cifs-client`, is required. See Section 3.2 for information about installing software components.
- USB memory dongle (see Section 3.6 for more information).
- Xmodem/Ymodem/Zmodem (use **rz/rx/rb/sz/sx/sb** commands from the management console). You can install these commands from software package `rzsz`. See Section 3.2 for information about installing software components.
  Note: The management console is not available for Access Point, only for Access Server.

For examples of transferring files, see Section 6.3.4.

2.4. Introduction to Configuration

When Access Server or Access Point is installed and powered up for the first time, the default configuration settings are being used. With these settings, the unit automatically configures its network settings assuming that it is connected to a LAN network with a DHCP server running. Additionally, the unit also uses Zero Configuration Networking (also known as Automatic Private IP Addressing) to connect to the 169.254.x.x network, which can be used if the network has no DHCP server.

After booting up, the only Bluetooth profiles enabled are the Object Push and File Transfer Profiles, used to send files to/from Access Server.

More Bluetooth profiles can be enabled, and most of Access Server and Access Point settings can be configured by using the **setup** application. It has a WWW interface at http://wrap-ip/setup but it can also be run at the command line.

All configurable settings in the **setup** application are listed in Appendix B with short help texts.

Note: The default username is **`root`** and the default password is **`buffy`**.

2.5. Using the Setup WWW Interface

The easiest way to change Access Server or Access Point settings is to use the WWW interface. Accessing the WWW interface is instructed in Section 2.2.

A typical WWW configuration page is shown in Figure 2-9 (This page can be found at Setup ⟶ Security settings)

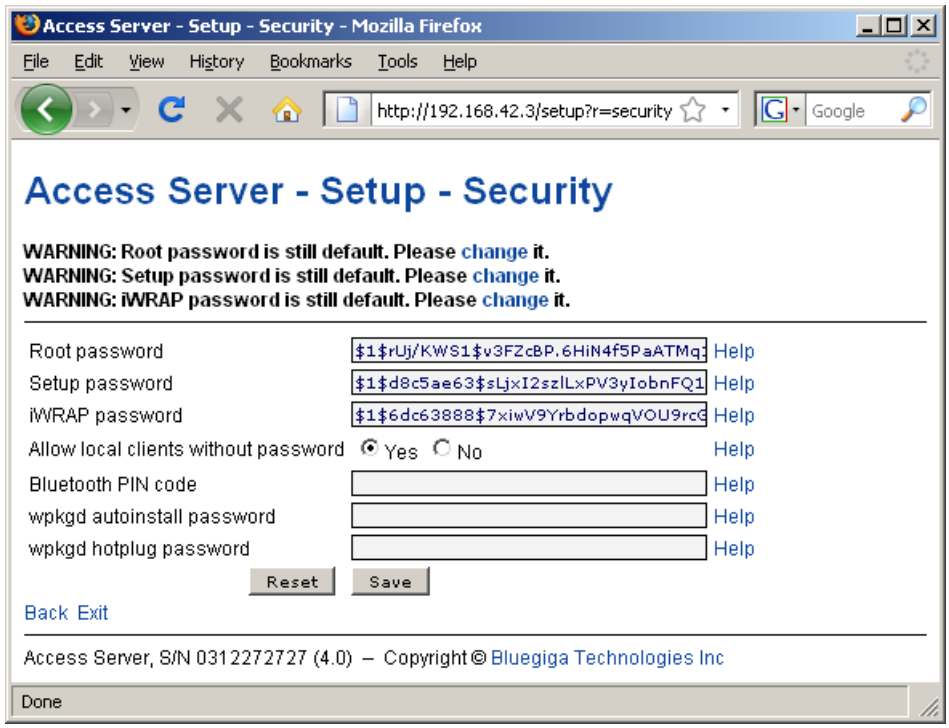

Figure 2-9. Example WWW Setup Page

The different parts of the WWW Setup page are discussed in the following list:

- Status area

  The status area serves three purposes:

  - It indicates that the changes are permanently saved when the user clicks the Save button (or when the user clicks a toggling Yes/No link).
  - It warns when default passwords are in use.
  - If invalid values were entered in one or more fields, an error message is shown in this area (see Figure 2-10).

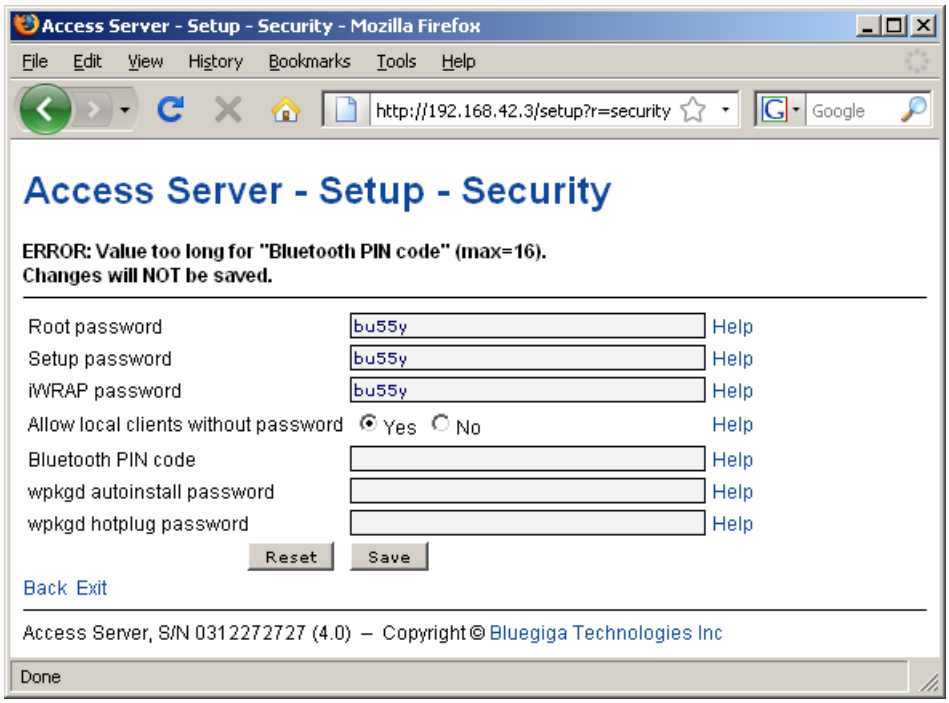

Figure 2-10. Trying to Save an Invalid Input

Note: It is typically necessary to reboot Access Server or Access Point for the changes to take effect. This can be done through the WWW interface (Advanced settings menu).

- Number or text entry fields

  Most of the configurable settings are text (or number) entry fields. For some fields, such as the IP address or netmask, there are restrictions on the input format. Setup validates the input at save time and accepts valid data only. The fields with errors are shown to the user so that mistakes can be fixed (see Figure 2-10).

- Help link

  Click the Help link to retrieve the setup page again with requested help information displayed. For an example, see Figure 2-11.

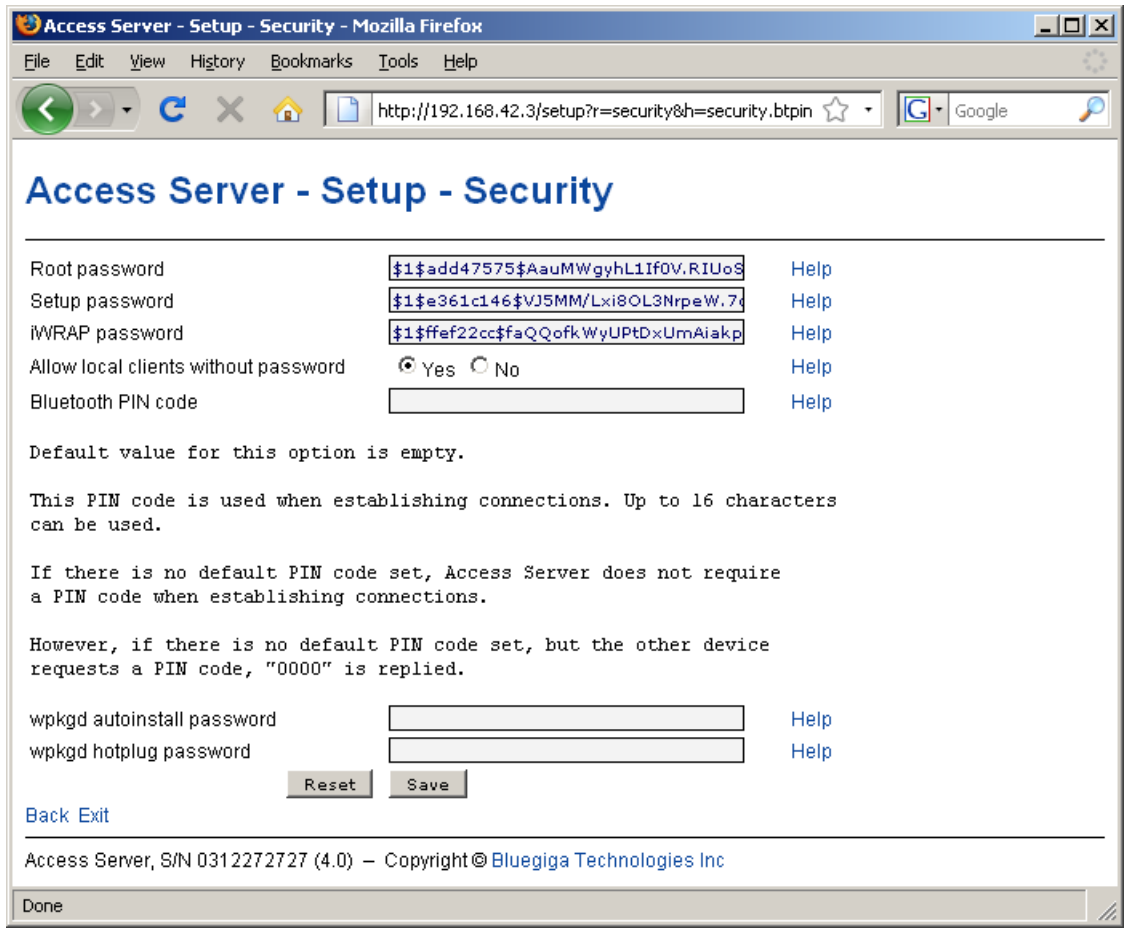

Figure 2-11. Help Links in WWW Setup

Note: If you have made changes to the settings on the page before clicking Help and not saved them yet, a warning message is displayed. See Figure 2-12

- Yes and No radio buttons

  These buttons are typically used to configure a setting that can be either enabled or disabled, and this setting has no effect on the visibility of other settings.

- Reset button

  Reset button resets the fields to the values currently in use at Access Server or Access Point. In other words, the Reset button discards unsaved changes.

  Note: The Reset button does *not* make a "factory reset".

- Save button

  Save button sends the WWW page to the setup application for validation. If the values in the fields are valid, they are permanently saved and the page is refreshed with the Changes have been saved. message at the top. The accepted values are shown in the page fields.

  If there were errors in the fields, these are shown as in Figure 2-10.

  Note: It is typically necessary to reboot Access Server or Access Point for the changes to take effect. This can be done through the WWW interface (Advanced settings menu).

- Back link

Press the Back link to return to the previous level of the Setup menu hierarchy.

Note: Pressing the Back link does *not* save changes in the fields on the current page. Therefore, if there are changes not saved, a warning message is displayed, see Figure 2-12.

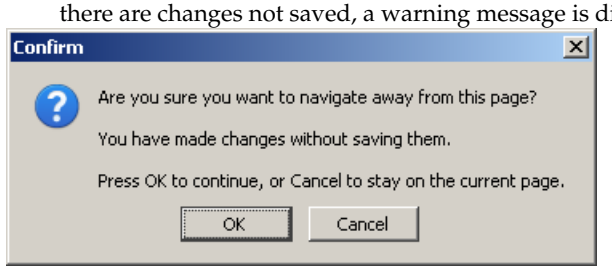

Figure 2-12. Warning Message of Changes not Saved

- Exit link

Exit link quits the setup application and returns to the Access Server's main WWW page.

Note: Pressing the Exit link does *not* save changes in the fields on the current page. Therefore, if there are changes not saved, a warning message is displayed, see Figure 2-12.

- Link to a configuration file

Some of the configurable settings are actually editable configuration files, such as `/etc/rc.d/rc.local` for Setup ⟶ Advanced settings ⟶ System startup script. Clicking the link will retrieve the file for editing in the browser window, or create a new file, if it does not exist. See Figure 2-13.

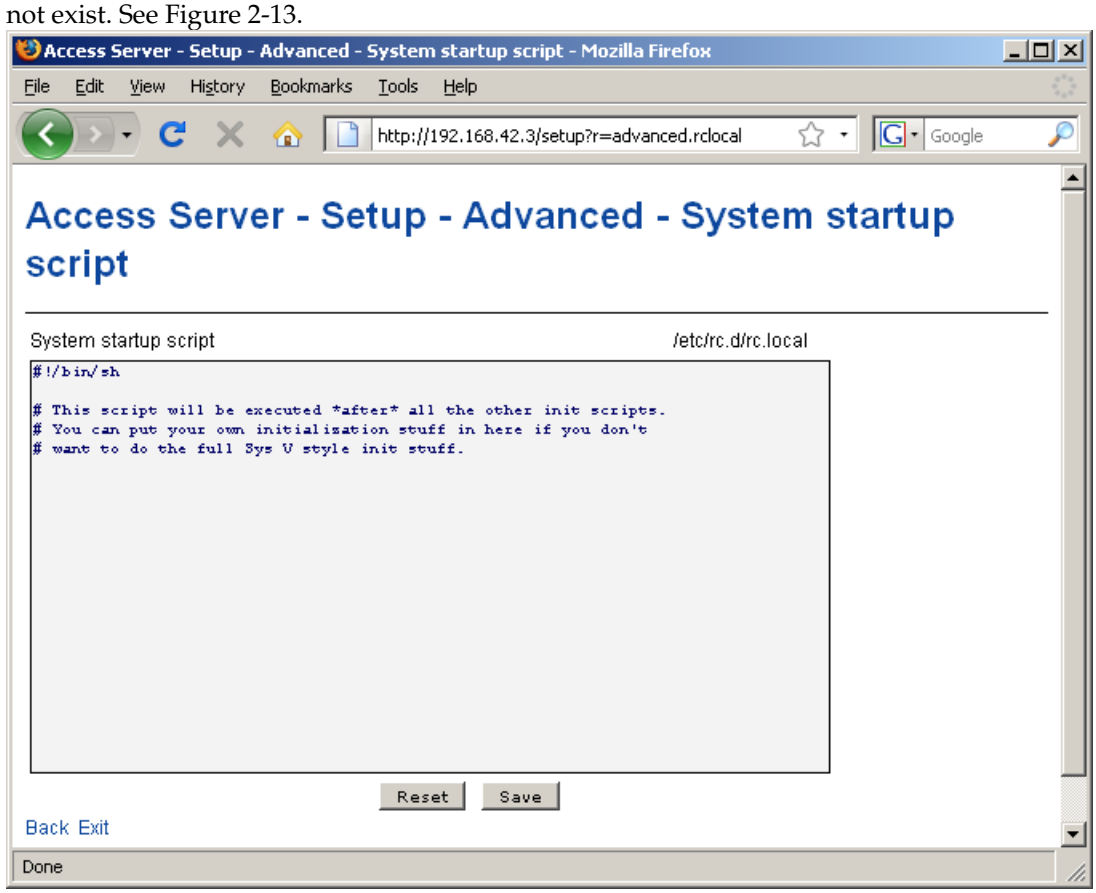

Figure 2-13. Editing Files in WWW Setup

Note: You can edit any file through the WWW Setup. to edit files, navigate to Setup ⟶ Advanced settings ⟶ Edit other configuration files.

- Toggling Yes/No and on/off links

  Clicking the Yes/No link (see Figure 2-14) immediately changes the setting and saves the change. Typically these links are used to display or hide further settings.

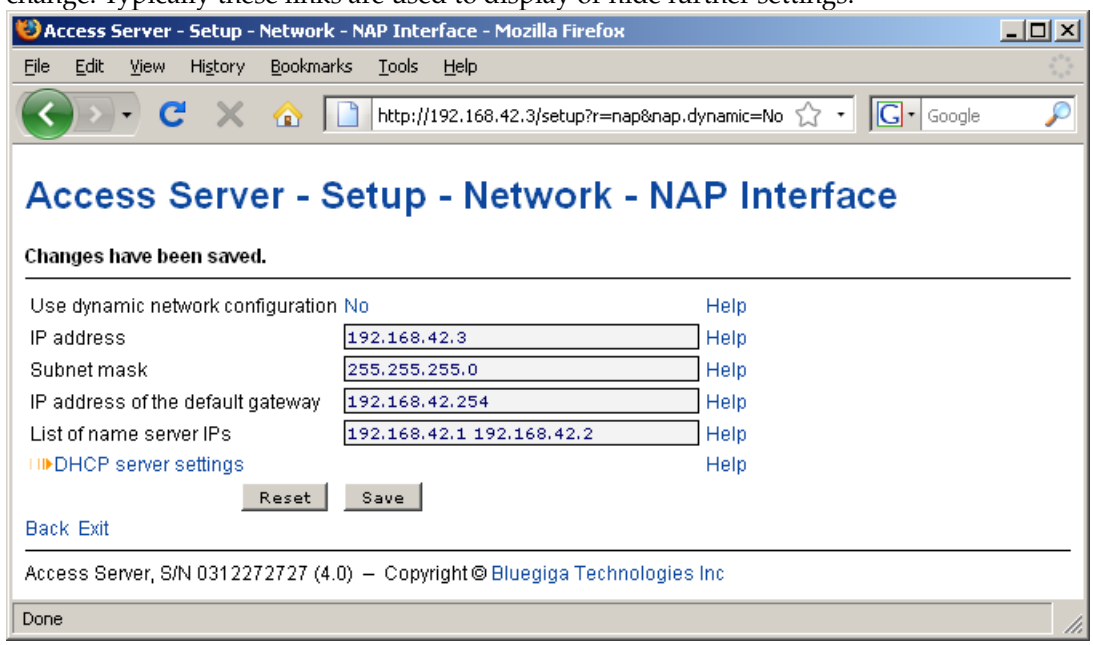

Figure 2-14. Yes / No links in WWW Setup

The on/off links in Setup ⟶ Applications ⟶ Default startup applications behave in a same way, making and saving the change immediately (see Figure 2-15).

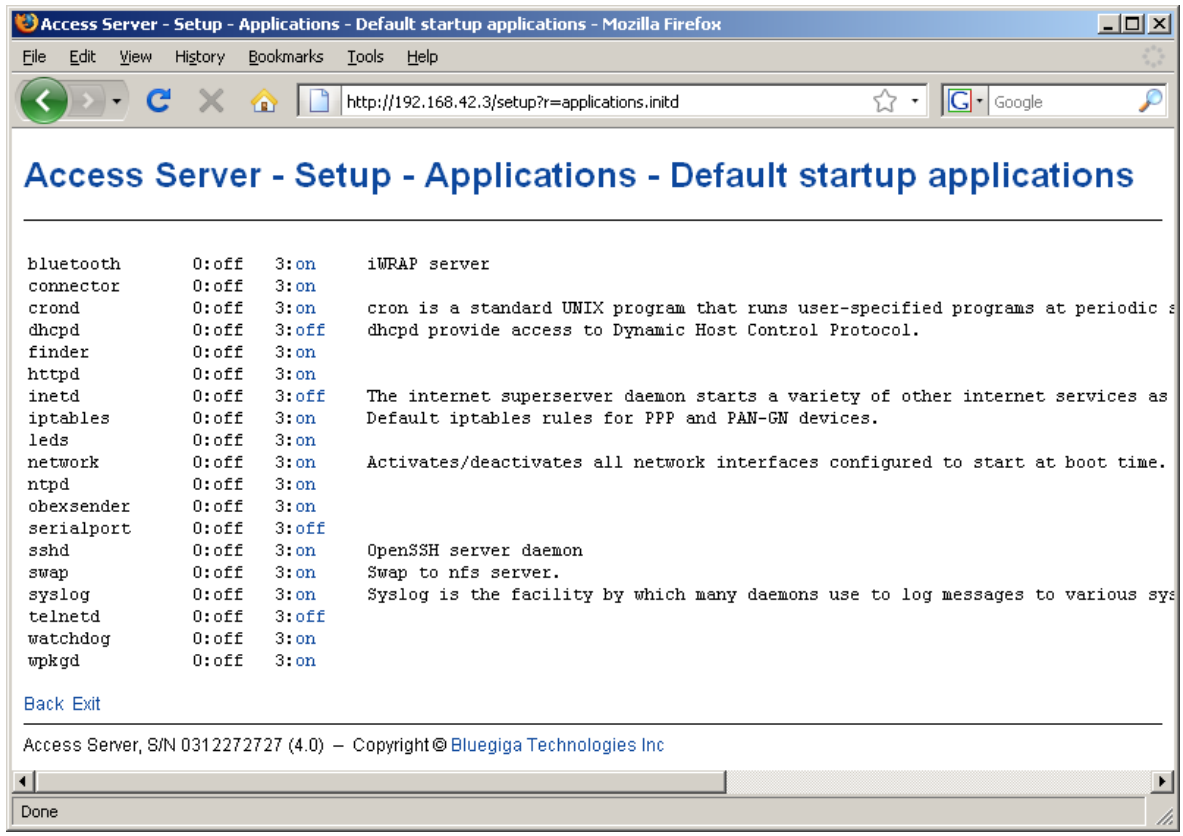

Figure 2-15. Selecting Default startup applications in WWW Setup

Note: To configure the default startup applications from the command line, use the **chkconfig** command.

- Upload links

The WWW Setup has settings that allow user to upload files to Access Server or Access Point, for example Setup ⟶ Advanced settings ⟶ Upload a software update (see Figure 2-16).

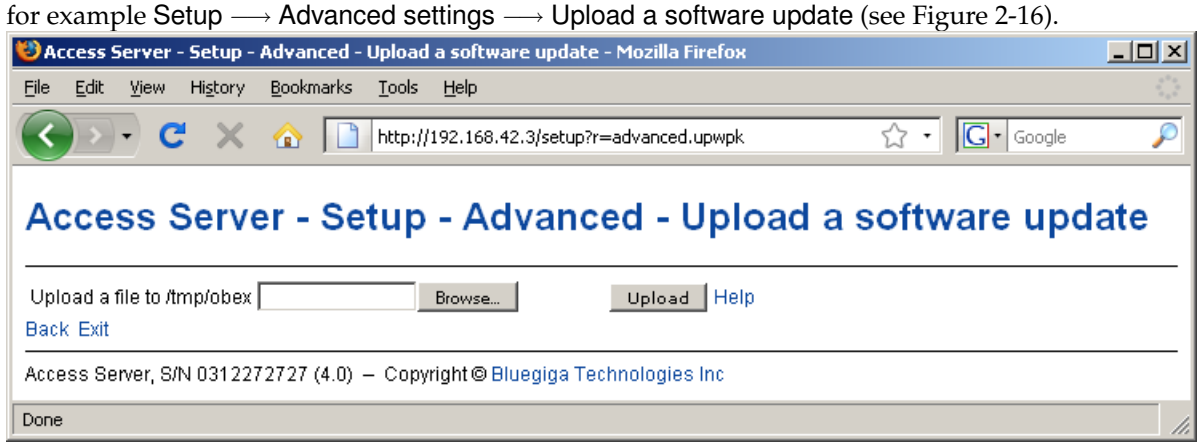

Figure 2-16. Uploading files through WWW Setup

Use the Browse... button to select the file to be uploaded, and send it to Access Server or Access Point by clicking Upload.

- Browsing files

  Some WWW Setup pages allow users to browse the file system or part of it, such as Setup ⟶ Advanced settings ⟶ Browse all files (see Figure 2-17).

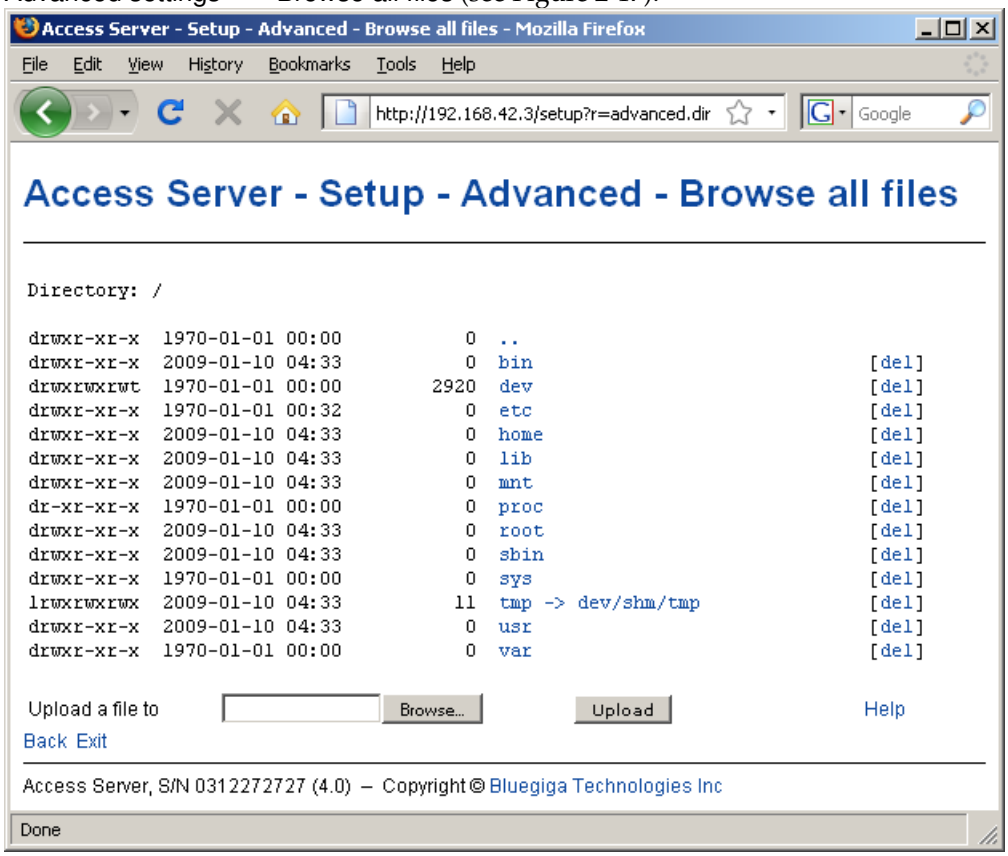

Figure 2-17. Browsing files via WWW Setup

Click the directory names to navigate in the file system.

Click a file name to view its contents.

Note: Currently it is only possible to view contents of text files. It is not possible to download files, either.

Click del to delete a file or an empty directory.

Warning

You will not be asked for confirmation for the deletion.

The WWW Setup also has menu items that run commands in Access Server or Access Point, and show the output in the browser window. Some commands, such as rebooting the unit, require your confirmation before execution.

2.6. Using the setup Command Line Application

The basic configuration settings can also be changed by using the **setup** application at the command line interface.

The **setup** application displays the settings in a hierarchical menu (see Figure 2-18). Navigating the menu is accomplished by entering the number or letter corresponding to the setting to be viewed and/or changed and pressing **Enter**. Pressing only **Enter** either accepts the previous value of the setting or returns to the previous level in the menu hierarchy.

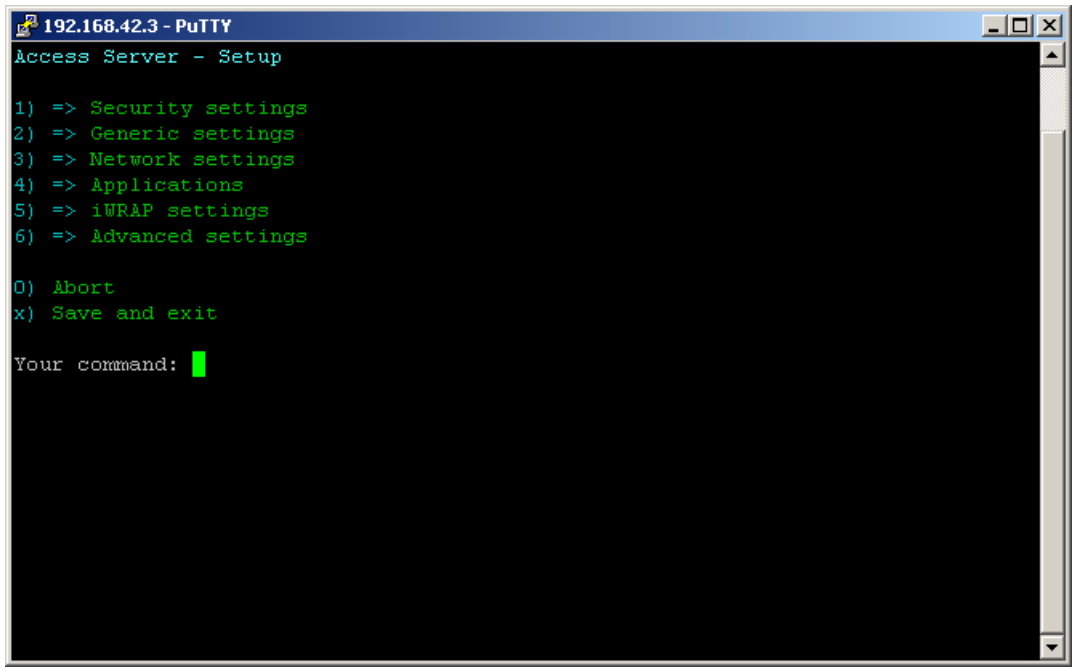

Figure 2-18. Using the setup Command Line Application

Note: Ensure that your terminal application does not send line ends with line feeds. If your terminal sends both CR and LF when you press **Enter**, you cannot navigate in the **setup** application.

If you use the command line setup application, files are edited by default with the **vi** editor. If you want to use another editor, you can specify it in `VISUAL` environment variable.

You can not enable or disable default startup applications using the command line setup application. Use command **chkconfig *application* on|off** instead.

2.7. Resetting a Configuration

You can reset the default configuration with the **setup -r** command. The command requires rebooting of Access Server or Access Point. When the system starts up, the default configuration settings are restored. If you have only changed the configuration by using the **setup** application, the following commands at the Access Server's or Access Point's command prompt will suffice:

```
[root@wrap /]$ setup -r
[root@wrap /]$ reboot
```

Note: This does not reset the edited files to factory defaults; it only affects the settings changed through the WWW Setup or the **setup** command line application. It does not uninstall extra applications installed by the user or reset Default startup applications settings, either. See Section 3.15 on how to perform a full factory reset.

2.8. Exporting and Importing Configurations

You can export configuration settings (expect for passwords and the list of default startup applications) by using the following command:

```
[root@wrap /root]$ setup -o > settings.txt
```

The saved settings can later be restored by using the following commands:

```
[root@wrap /root]$ setup -m settings.txt
[root@wrap /root]$ reboot
```

Chapter 3. Using the System

This chapter describes the basic features of Bluegiga Access Server and Access Point. This includes information on using the device as a Bluetooth LAN/PAN Access Point or a Bluetooth Serial Port, using the Web Server, ObexSender, and WRAP Package Management System. The various ways of uploading content for browsing and/or downloading are also included, as well as getting familiar with the utility applications and services.

Using the features described in this chapter does not require Bluegiga Software Development Kit to be installed.

Note: The default username is **`root`** and the default password is **`buffy`**.
Note: Most of the configuration files are in Linux text file format, where the lines end with a single Line Feed (LF, "\n") character. Some applications will not work if the configuration file format is changed to MS-DOS format (this happens, for example, if you transfer the files to Windows for editing with Notepad), where the lines end with both Carriage Return and Line Feed (CR+LF, "\r\n") characters. Remember to convert the files to "UNIX" format before transferring them to Access Server or Access Point.

3.1. Network Interfaces

The Access Server and Access Point network interfaces are described in Table 3-1.

Interface	Description
nap	Dynamic virtual ethernet ("cable") device. This is the device having an IP address. All the programs should use this device instead of eth0.
nap:9	Alias interface of `nap` for zero configuration networking.
eth0	The real ethernet device, which is dynamically linked to the nap device. Do not use this device, use nap instead.
wlan0	Wi-Fi device. In the client mode (default), this device has its own IP address. In the access point mode, it is dynamically linked to the nap device (the default interface).
wifi0	Virtual control device for wlan0. Do not use this device.
gn	Virtual device for Bluetooth PAN-GN connections.
bnep#	These devices are used for incoming and outgoing Bluetooth PAN connections. These devices are created, deleted and linked (to nap or gn) dynamically.
ppp#	These devices are used for incoming and outgoing LAP connections or for GPRS modem connection. In LAP use, these devices are created and deleted dynamically and traffic coming from them is masqueraded to the nap device. When GPRS modem is enabled, all traffic to ppp interfaces is also masqueraded.
lo	Local loopback interface.

Table 3-1. Access Server and Access Point Network Interfaces

3.2. Managing Software Components

To maximize memory available for customer applications, Access Servers and Access Points

ship with minimal amount of software components installed. To see the installed software components and their version numbers, navigate to Setup ⟶ Advanced ⟶ System Information ⟶ List installed software components or give command **wpkgd -l** at the shell prompt. See Appendix C for more information of software components installed by default and available separately.

3.2.1. Software Component Package Naming

Software components are delivered in package files which are named in format `name-[version].[architecture].wpk`. For example: `smsgw-20080910-1.lt.wpk` is `smsgw` software component, version number `20080910-1`, for `lt` architecture. You can only install a software package of a specific architecture to hardware that supports the architecture. The architectures and the supporting hardware are listed in Table 3-2

Architecture	Supporting Hardware
noarch	Any Access Server or Access Point
lt	Access Point 3201
if	Access Server 229x with Serial Number 0607240000 or higher
df	*Not supported since software version 4.0*. Access Server 229x with Serial Number 0607239999 or lower (old non-RoHS Access Servers)

Table 3-2. Software Component Package Architectures

3.2.2. Installing Software Components

If you can access the command prompt of your Access Server or Access Point and your device has access to Internet, you can manage software components using network update operations described in Section 3.2.4. You can also install software components by inserting a USB dongle with the WPK file containing the software installation packet in its root directory or by transferring the WPK file to `/tmp/obex` directory on Access Server or Access Point. The easiest way to transfer a WPK to this directory is to upload it from WWW Setup at Setup ⟶ Advanced settings ⟶ Upload a software update.

The WPK files of additional software component provided by Bluegiga can be found in following locations:

- http://update.bluegiga.com/as/[version]/[architecture], where `[version]`is the software release like 4.0 and `[architecture]` either `lt` or `if` (see Table 3-2)
- `wpk`-directory in Bluegiga SDK DVD-ROM or ISO image
- After Bluegiga SDK is installed, in corresponding application and library directories under `asdk`-directory

3.2.3. Uninstalling Software Components

You can uninstall software components from the shell prompt. To list installed software components use command **wpkgd -l**. To uninstall a component, use command **wpkgd -e [component]**. See the **wpkgd** command without parameters for more information.

3.2.4. Network Update Operations

When Access Server or Access Point is connected to the Internet, you can use the **wpkgd** command at the shell prompt to easily manage installed software components:

wpkgd install *pkg*

Install the newest available software component called *pkg* for your architecture. The software is retrieved from the Bluegiga software update repository. Example:

```
[root@wrap root]$ wpkgd install smsgw
Downloading http://update.bluegiga.com/as/4.0/lt/smsgw-20090910-1.lt.wpk
Package "smsgw" installed
```

> **Note:** Currently the command **wpkgd install *pkg*** may report that software is installed even if the installation has failed because of failing dependencies. It is therefore worth ensuring that the installation has been successful, by using command **wpkgd search *pkg***

wpkgd erase *pkg*

Remove a software component called *pkg*. Example:

```
[root@wrap root]$ wpkgd erase smsgw
Purging smsgw (20080910-1)...
```

wpkgd update

Update all installed software components.

wpkgd update *pkg*

Update the software component called *pkg*.

wpkgd list

List installed and available software components and their version numbers.

wpkgd search *keyword*

Search for software component packages with name matching *keyword*.

wpkgd clean

Clean network cache.

3.3. Bluetooth

The iWRAP servers (one server in Access Server 2291 and Access Point 3201, two in Access Server 2292 and three in Access Server 2293) are automatically started at power-up. By default, the Object Push and File Transfer Profiles are activated. The iWRAP servers can be accessed and controlled (by applications or even interactively with a telnet client) through the iWRAP interface, described in Chapter 7. Currently, there can be up to 14 simultaneous Bluetooth connections between a single master iWRAP server and up to seven simultaneous slaves.

3.3.1. iWRAP Password Protection

The access to iWRAP can be password protected. The default password is **buffy**, but it can be set off or changed with the **setup** application (see Section 2.4). The password is case sensitive. The password must be typed in as the first command after the server has replied with "READY."

3.3.2. LAN Access Profile

This profile is not automatically started at boot. The default settings can be changed with the **setup** application (see section Section 2.4), or runtime with the iWRAP interface (see Chapter 7).

Access Server or Access Point can also act as a LAN Access Client, but in this case it must be controlled manually using iWRAP commands, as described in Chapter 7.

Note: As of Bluetooth specification 1.2, LAN Access Profile has been deprecated.

3.3.3. Serial Port Profile

This profile is not automatically started at boot. The default settings can be changed with the **setup** application (see section Section 2.4).

The Serial Port Profile is used to replace an RS-232 serial cable between two devices with a Bluetooth connection. The physical setup is shown in Figure 3-1.

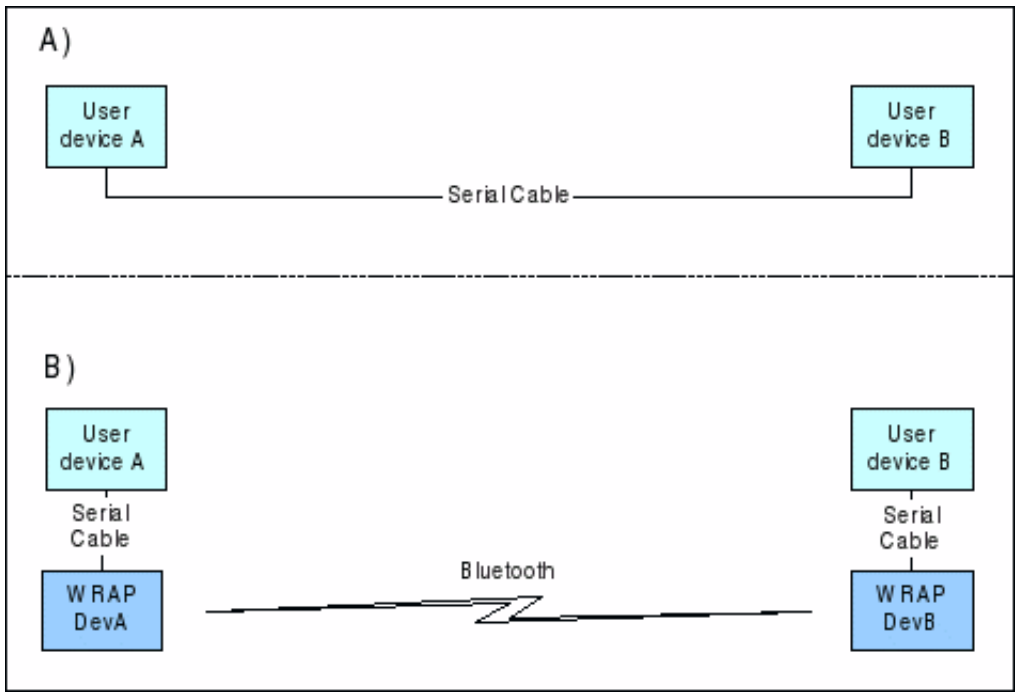

Figure 3-1. Serial Cable Replacement Physical Setup

State A) in the figure is the starting situation with a serial cable connecting the devices. This cable is to be replaced with a Bluetooth connection.

In state B) the long serial connection is replaced with a Bluetooth Serial Port Profile connection between the two Access Server devices. These devices are then locally connected to the user devices with (short) serial cables (in case of Access Point without a physical serial port, a USB-to-Serial converter device is needed). The cable between user device A and Access Server device A must be a cross-over cable. The cable between user device B and Access Server device B must be similar (direct or cross-over) to the one used in state A).

If RTS/CTS handshaking is used to ensure correct data transfer, the serial cables must have these pins connected. Notice that this handshaking is "local": it takes place between the user device and Access Server. No handshaking between user device A and user device B on the other end of the Bluetooth connection is provided.

If RTS/CTS handshaking is not used, CTS must be connected to DTR.

DCD, DTR, and DSR signals are not supported. This also means that user devices A and B will not be able to tell whether or not the Bluetooth connection is up.

When the physical setup is ready, you can configure the Bluetooth Serial Port Profile settings. By default, the profile is listening in DevB mode, at 115200 bps, 8 data bits, no parity, 1 stop bit, and RTS/CTS enabled. To change these settings, use the **setup** application or the WWW Setup interface, as described in Section 2.4.

> **Note:** To enable Serial Port Profile, navigate to Setup ⟶ Applications ⟶ Default startup applications in the WWW Setup interface, and switch the serialport application to on.
> Enabling can also be done from command prompt with command **chkconfig serialport on**.

3.3.4. Object Push and File Transfer Profile

Access Server and Access Point have two OBEX profiles, Object Push Profile (ObjP) and File Transfer Profile (FTP). You can use these profiles to transfer files between different Access Servers or Access Points and other devices supporting ObjP or FTP.

3.3.4.1. Incoming ObjP and FTP

Incoming ObjP and FTP connections are handled by forwarding the call to the **obexserver** program, which handles both profiles. By default the working directory is `/tmp/obex`. FTP users have full read and write access to that directory.

In ObjP mode, **obexserver** will prefix received files with sender's Bluetooth address and iWRAP port number. In the case of a duplicate filename, a counter is also appended to filename.

You can change the settings of **obexserver** in Setup ⟶ iWRAP settings ⟶ Bluetooth profiles ⟶ Object push profile settings. The working directory is configured with its own settings, other settings can be changed by altering the Optional parameters for server setting.

You can add parameter **--fork *cmd*** to run your command after each received file. The **--fork** parameter understands the following meta characters:

Meta	Description
$$	Character '$'
$r	Configured root directory
$p	Configured prefix
$b	Remote's Bluetooth address
$t	Temporary file name, with directory
$T	Real file name, without prefix and directory
$f	Real file name, with prefix and directory
$F	Real file name, with prefix, but without directory

Meta	Description
$d	Current UNIX timestamp

Table 3-3. obexserver's metas

If the **--fork**'ed program returns a non-zero errorlevel, the received file will be deleted.

Changing Optional parameters for server settings to contain the following example will save a copy of incoming files to `/tmp` directory with the name in format `configured_prefix-timestamp-filename`:

```
--bdaddr $b --prefix $b-$P- --fork '/bin/cp $$t /tmp/$$p$$d-$$T'
```

3.3.4.2. Outgoing ObjP and FTP

Three simple utilities, **obexput**, **obexget** and **obexsender-put** are provided. They can be used to send and retrieve files to and from another Bluetooth device using ObjP or FTP.

Usage:

```
obexput [parameters] bdaddr channel file(s)

obexget [parameters] bdaddr channel file(s)

obexsender-put parameters
```

Enter any of these commands without parameters to get a short help for using the command.

> **Note:** You can use keyword "OBJP" or "FTP" as the "channel" parameter for automatic device and service discovery.

For **obexput** and **obexget**, a non-zero return value indicates an error. The reason for this error is printed to standard output.

> **Tip:** You can use **obexput** easily from iWRAP (see Chapter 7) with following syntax:
> `CALL bdaddr OBJP FORK \"/usr/bin/obexput - 1 filename\"`
>
> Value `-` as bdaddr and `1` as channel tells **obexput** that it is launched by the iWRAP server, and that data connection is bound to standard input and output.

Originally **obexsender-put** was a helper application for ObexSender. It can also be used by other programs. It calls the given Bluetooth device, calculates a device hash value and then sends one or more files to it using ObjP. **obexsender-put** takes the following parameters:

Parameter	Description
--configfile name	Use "name" as the configuration file.
--bdaddr bd	Call to "bd" and send files specified in the configuration file.
--iwraphostname host	Use iWRAP in host. Defaults to "localhost".
--iwrapport port	Use iWRAP in port. Defaults to "10101".
--iwrappassword pass	Use iWRAP password. Defaults to empty.

Parameter	Description
--verbose level	Use the debug verbosity level. Defaults to "0". Logging is written to standard output.
--hash bd	Calculate and show hash value by calling to "bd", do not send anything.
--uuid uuids	Use specified UUIDs. Defaults to "OBEXOBJECTPUSH,OBEXFILETRANSFER".

Table 3-4. Parameters for obexsender-put

The configuration file specifies filenames and hash values for them. At first, there has to be one or more "regex" lines, followed by one or more "file" or "exec" lines, followed by an empty line. There can be multiple instances of these tuples. The syntax is:

```
regex <match1>
regex <match2>
file <fakename1> </path/to/file1>
file <fakename2> </path/to/file2>
exec </path/to/command1>
exec </path/to/command2>

regex <match3>
file <fakename3> </path/to/file3>

...
```

Parameter "match" is a regex of hash. If it matches with the calculated hash, the file(s) in this tuple will be sent. Parameter "fakename" specifies a name of the file shown to the receiver. See the `lottery` example in the SDK for more information about **exec**.

Example of a configuration file:

```
regex Nokia.9500
file hello.jpg /usr/local/obexsender/files/communicator.jpg

regex .
file hello.jpg /usr/local/obexsender/files/unknown.jpg
```

The possible return values of **obexsender-put** are shown in Table 3-5.

Value	Description
0	OK, files sent without errors.
1	Error in parameters.
2	Fatal error. System reboot is needed!
3	No files sent. Operation should be retried.
4	No files sent. Remote refused to receive, do not retry.
5	OK, no files matched the hash, nothing sent.

Value	Description
6	Error, file not found from disk.
7	Error, remote device does not support specified UUID(s).

Table 3-5. Errorlevels from obexsender-put

There is also another ObexSender helper application called **obexsender-inquiry**. It can be used to inquire for nearby Bluetooth devices. Its usage and return values are similar to the **obexsender-put** command. All results are written to standard output.

Usage:

```
obexsender-inquiry parameters
```

Parameter	Description
--pair-only	List all paired devices and exit.
--inquiry	Inquiry and exit.
--ready-check	Check if iWRAP is ready or not.
--iwraphostname host	Use iWRAP in host. Defaults to "localhost".
--iwrapport port	Use iWRAP in port. Defaults to "10101".
--iwrappassword pass	Use iWRAP password. Defaults to empty.
--verbose level	Use the debug verbosity level. Defaults to "0". Logging is written to standard output.

Table 3-6. Parameters for obexsender-inquiry

Value	Description
0	OK.
1	Error in parameters.
2	Fatal error. Reboot!

Table 3-7. Errorlevels from obexsender-inquiry

3.3.5. PAN Profiles

Access Server and Access Point have support for all PAN profile modes: Personal Area Network User (PANU), Network Access Point (NAP) and Generic Networking (GN). Accepting incoming PAN connections to any of these modes is disabled by default for security reasons.

Access Server and Access Point can be configured to accept incoming PAN connections and the default settings can be changed by using the **setup** application (see section Section 2.4).

The Network Access Point mode is the most useful PAN profile mode. You can enable it by sending the `enable-pan.noarch.wpk` file (available on-line at http://bluegiga.com/as/current/enable-pan.noarch.wpk) to Access Server or Access Point by using the Bluetooth Object Push profile. Alternatively, you can copy the file to the root of a USB memory dongle and insert the dongle to device's USB port.

The device creating the PAN connection decides upon the modes to be used. Access Server or Access Point automatically handles incoming connections. Access Server or Access Point can also act as a PAN client, but in this case it must be controlled manually by using the iWRAP interface, described in Chapter 7.

3.3.6. Changing the Bluetooth Range

The transmit power of Access Server or Access Point is configurable. By default, class 1 (100 meter range) settings are used. The settings can be changed down to "class 2" (10 meter range) settings with the **btclass 2** command, or even lower with the **btclass 3** command. Class 1 settings can be restored with the **btclass 1** command. You can also find these commands in Setup ⟶ Advanced settings ⟶ Bluetooth commands menu in the WWW Setup interface.

After **btclass #** is given, it is recommended to reboot Access Server or Access Point once to restart ObexSender and other applications connected to the iWRAP server(s).

Note: It is recommended to stop all applications using Bluetooth before issuing the **btclass** command.

3.3.7. btcli

You can send iWRAP commands from the command line by using the **btcli** application. See Section 7.4 for more information.

3.4. GSM/GPRS/3G Modems

Access Server and Access Point can be connected to Internet over GPRS/3G using USB modems from several vendors. Access Server 229x can also connect to Internet using a GSM/GPRS Compact Flash card or an external modem connected to its serial port. The supported devices are listed in Appendix E. Using these devices, Access Server and Access Point can also act as an SMS gateway to send and receive SMS messages.

3.4.1. Enabling Modem Support

Compact Flash GPRS modems and modems connected to Access Server's serial port are supported without additional software. Support for USB modem devices is available in software component `kernel-modules-modem`, which is installed by default. See Section 3.2 for information about installing and updating software components. The operating system automatically identifies supported USB or Compact Flash GPRS devices and loads correct drivers when they are inserted.

3.4.2. Using GPRS

You can enable the GPRS modem and configure its settings, such as the modem device and connection script details, by using the setup application or its WWW interface. For more information, see Section 2.4 and documentation for Setup ⟶ Network settings ⟶ Enable modem interface in Appendix B.

Note: A reboot is needed for the new settings to take effect. From WWW Setup, you can do this at Setup ⟶ Advanced settings ⟶ Reboot system (confirm).

When GPRS modem is enabled, by default Access Server or Access Point tries to establish GPRS Internet connection only once when the device boots up. It is therefore recommended to enable option Setup ⟶ Network settings ⟶ Modem settings ⟶ Force connection open. With this enabled, GPRS Internet connection is checked every 10 minutes with the **ping** command. If the check fails, GPRS connection is restarted.

Note: By default, Force connection open uses host 194.100.31.45 (bluegiga.com) for checking that the GPRS connection is working. You might want to specify a reliable host closer to your system in Setup ⟶ Network settings ⟶ Modem settings ⟶IP address used in force check. The test host must respond to ICMP ECHO_REQUEST packets generated by **ping** command, otherwise the GPRS connection is reset.
Note: If you also want to use the ethernet connection, you must remove it from the default interface (`nap`) bridge and configure its network settings individually using the **setup** application while keeping the default interface network settings in their default (dynamic) state.

3.4.3. Using SMS Gateway Server

Bluegiga SMS Gateway Server supports Nokia 20, Nokia 30, or Wavecom WMOD2 compatible GSM terminals and the supported GSM/GPRS Compact Flash or USB modems for sending and receiving SMS messages.

Bluegiga SMS Gateway Server is not installed by default. It can be installed from software component `smsgw`. See Section 3.2 for more information about installing software components.

When Bluegiga SMS Gateway Server is installed, it is also enabled to start at boot by default. You can disable it later (for example if you need to use the same modem for GPRS Internet connection) either with command **chkconfig smsgw off** or using the **setup** application's WWW interface, as described in section Section 2.4. Disabling and enabling is done at Setup ⟶ Applications ⟶ Default startup applications ⟶ smsgw.

By default, Bluegiga SMS Gateway Server assumes the modem can be accessed using `/dev/ttyUSB0` device. The device can be changed by using the **setup** application or its WWW interface, by changing the setting at Setup ⟶ Applications ⟶ SMS gateway settings ⟶ Modem device. Another mandatory setting is the SMSC (Short Message Service Center) number. Remember to change it to match your mobile operator.

Note: A reboot is needed for the new settings to take effect. From WWW Setup, you can do this at Setup ⟶ Advanced settings ⟶ Reboot system (confirm).
Note: The PIN code query of the SIM card at power-up must be disabled.

Ensure you have GPRS modem disabled at WWW Setup ⟶ Network settings ⟶ Enable modem interface ⟶ No. Remember that rebooting is needed when setup settings are changed.

Note: Bluegiga SMS Gateway Server requires exclusive access to the GPRS modem device. Otherwise it will fail to start and the `can't lock device [devicename]` error message is printed to the system log. Especially, if you are using Bluetooth Serial Port Profile, ensure it is configured to use another serial port device or disabled completely.
Note: To use Nokia terminals, the device must be connected to the user serial port when the server starts up. Also, the terminal must be configured to operate in RS-232/AT command. Nokia terminals are configured with the N20 or N30 Configurator application.

By default, Bluegiga SMS Gateway Server uses directory `/tmp/sms/in` for storing incoming messages (each message received is stored in a separate file). It scans messages to be sent from directory `/tmp/sms/out`. These settings can be changed by editing the configuration file at Setup ⟶ Applications ⟶ SMS gateway settings ⟶ Edit configuration file (search for `dirin` and `dirout` entries.

Warning

Bluegiga SMS Gateway Server exits in case of error. As it has registered itself to Bluegiga User Level Watchdog, this will make device to reboot. This is a feature to recover from problems in modem communication, but as a side effect it can cause a reboot loop if there is a mistake in the configuration file. Be careful when editing it.

To send a SMS message, create a text file with extension `.sms`. The first line of that file must contain only the GSM number of the recipient. Next lines contain the message. After you have created the file, copy or move it to the outgoing directory (`/tmp/sms/out` by default) and the message will be sent automatically.

An example message:

```
+17815550199
Hello, world!
```

Once the message is sent, the file is deleted from the outgoing directory.

For further information on using **smsgw**, see the **makesms** example in Section 6.3.1.

3.5. Compact Flash and USB Wi-Fi devices

Access Server and Access Point support several USB Wi-Fi dongles. Access Server also supports Prism II/III based Compact flash Wi-Fi cards. The supported devices are listed in Appendix E.

Note: Ad hoc mode is not supported.
Note: WPA/WPA2 passphrase length must be 8..63 characters.

3.5.1. Using Wi-Fi as Client (Managed)

USB Wi-Fi client device support (including `wpa-supplicant` for WPA or WPA2 encryption support) is installed by default. Install software component `kernel-modules-hostap` for Compact Flash Wi-Fi client device support. See Section 3.2 for information about installing software components.

Enable Wi-Fi interface using the setup application or its WWW interface at Setup ⟶ Network settings ⟶ Enable Wi-Fi interface.

When the correct kernel modules are installed and Wi-Fi interface enabled, Access Server or Access Point notices when a supported Wi-Fi card is inserted and tries to use it in the client mode, without encryption. So, if there is an open Wi-Fi access point in range, you will automatically connect to it.

To change Wi-Fi settings, use the setup application or its WWW interface at Setup ⟶ Network settings ⟶ Wi-Fi settings.

When using WEP or no encryption, configure ESSID to match the ESSID of the Wi-Fi access point to connect and WEP encryption keyto match the WEP encryption key used to connect, or empty if encryption is not used.

When using WPA or WPA2, change Use WPA encryption to yes. Furthermore, use the Edit WPA configuration file menu option and modify the configuration file's rows with "ssid" and "psk" so that "ssid" matches the ESSID and "psk" matches the password of your configuration.

Note: A reboot is needed for the new settings to take effect. From WWW Setup, you can do this at Setup ⟶ Advanced settings ⟶ Reboot system (confirm).
Note: The current software version does not support Wi-Fi bridging in the client (managed) mode, which means that traffic from Wi-Fi cannot be forwarded to wired ethernet.

To debug Wi-Fi issues, you can collect and review Wi-Fi diagnostics information using setup application or its WWW interface at Setup ⟶ Network settings ⟶ Wi-Fi settings ⟶ Collect Wi-Fi diagnostics.

In addition, a standard set of command line wireless utilities is provided to fine-tune your Wi-Fi configuration:

- **iwconfig**
- **iwlist**
- **iwpriv**

For more information on these utilities, see: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.h

3.5.2. Using Wi-Fi as Access Point (Master) with USB Dongles

Please refer to Appendix E to see which USB Wi-Fi dongles support access point mode. Install, in this order, software packages `libnl` and `hostapd`. See Section 3.2 for information about installing software components.

Enable Wi-Fi interface using the setup application or its WWW interface at Setup ⟶ Network settings ⟶ Enable Wi-Fi interface.

To change Wi-Fi settings, use the setup application or its WWW interface at Setup ⟶ Network settings ⟶ Wi-Fi settings.

To begin with, change the setting Act as a Wi-Fi Access Point to yes and Use hostapd to yes (even if you are not using encryption or you are using WEP).

Then, use Edit hostapd configuration file and uncomment and modify the following lines:

```
# nl80211 driver interfaces require bridge parameter
bridge=nap
# driver interface type
driver=nl80211
# broadcasted ESSID:
ssid=MyAccessPoint
# two letter country code where access point is located:
country_code=FI
# USB dongles are g-devices:
hw_mode=g
# Wi-Fi channel (anything between 1 and 11 should be safe choice):
channel=6
```

Then, depending on used encryption, change the following lines:

No encryption

Change nothing more.

WEP encryption

```
wep_default_key=0
            wep_key0="mysecretkey"
```

WPA-TKIP

```
wpa=1
            wpa_passphrase=verysecretpassphrase
```

WPA2-TKIP

```
wpa=2
            wpa_passphrase=verysecretpassphrase
```

WPA-TKIP and WPA2-TKIP

```
wpa=3
            wpa_passphrase=verysecretpassphrase
```

WPA-CCMP (AES)

```
wpa=1
            wpa_passphrase=verysecretpassphrase
            wpa_pairwise=CCMP
```

WPA2-CCMP (AES)

```
wpa=2
            wpa_passphrase=verysecretpassphrase
            rsn_pairwise=CCMP
```

WPA-CCMP and WPA2-CCMP (AES)

```
wpa=3
            wpa_passphrase=verysecretpassphrase
            wpa_pairwise=CCMP
            rsn_pairwise=CCMP
```

3.5.3. Using Wi-Fi as Access Point (Master) with Compact Flash Cards

Please refer to Appendix E to see which Compact Flash cards are supported. Install, in this order, software packages `kernel-modules-hostap`, `libnl` and `hostapd`. See Section 3.2 for information about installing software components.

> **Note:** Older Compact Flash cards with firmware version 1.4.2 or earlier do not work in the access point mode. Instead, you will see an error message in the system log (`/var/log/messages`, viewable at Setup ⟶ Advanced settings ⟶ System Information ⟶ Show system log file).

To change Wi-Fi settings, use the setup application or its WWW interface at Setup ⟶ Network settings ⟶ Wi-Fi settings.

First, change the setting Act as a Wi-Fi Access Point to yes and ESSID to ESSID you want to broadcast.

If you want to use no encryption, you are done. If you want to use WEP encryption, edit WEP encryption key.

If you want to use WPA encryption (or hostapd extended functionality), change Use hostapd to yes and use Edit hostapd configuration file and uncomment and modify following lines:

```
# hostap driver does not need bridge parameter
# bridge nap
```

```
# driver interface type can be left uncommented as well
# driver=hostap
# broadcasted ESSID:
ssid=MyAccessPoint
# two letter country code where access point is located:
country_code=FI
# CF cards are b-cards
hw_mode=b
# Wi-Fi channel (anything between 1 and 11 should be safe choice):
channel=6
```

Then, depending on encryption used, change the following lines:

WPA-TKIP

```
wpa=1
            wpa_passphrase=verysecretpassphrase
```

WPA2-TKIP

```
wpa=2
            wpa_passphrase=verysecretpassphrase
```

WPA-TKIP and WPA2-TKIP

```
wpa=3
            wpa_passphrase=verysecretpassphrase
```

WPA-CCMP (AES)

```
wpa=1
            wpa_passphrase=verysecretpassphrase
            wpa_pairwise=CCMP
```

WPA2-CCMP (AES)

```
wpa=2
            wpa_passphrase=verysecretpassphrase
            rsn_pairwise=CCMP
```

WPA-CCMP and WPA2-CCMP (AES)

```
wpa=3
            wpa_passphrase=verysecretpassphrase
            wpa_pairwise=CCMP
            rsn_pairwise=CCMP
```

Note: The current driver cuts all Wi-Fi connections if one client tries to connect with an invalid key. This happens only with Compact Flash cards when using WPA/WPA2. Connections will resume when clients reconnect.

3.6. USB Storage Devices and Compact Flash Memory Cards

Access Server's and Access Point's persistent memory storage can be extended by using a USB storage device like a memory dongle or a portable hard drive or a Compact Flash memory card. These are also used by the Bluegiga Remote Management System (see Section 3.9.4) - each time this kind of device is inserted, it is automatically mounted and scanned for management packets, which are processed and unmounted.

To use the USB storage device or Compact Flash memory card for your own applications, the memory must be mounted manually by using command:

```
[root@wrap /]$ mount -t vfat device directory
```

The *device* parameter is a path to the USB dongle or Compact Flash memory card filesystem device. For the first memory device inserted after a reboot, it is `/dev/sda1` if the device is partitioned (which often is the case), or `/dev/sda` if the device has no partition table. If you insert more memory devices at the same time, new device file names are created: `/dev/sdb1` for the second one, `/dev/sdc1` for the third one, and so on. If you unmount and remove the first memory device before inserting the second one, new device file names are not created.

Note: Always remember to unmount the memory dongle or memory card with command:

```
[root@wrap /]$ umount directory
```

Note: If you have inserted both a USB memory dongle and a Compact Flash memory card before powering up Access Server, Compact Flash card is found first (typically getting device file name `/dev/sda1`) and the USB memory device is found next (`/dev/sdb1`).

The filesystem in USB dongle can get corrupted if you have a power failure while you are writing data to it. A utility called **fsck.vfat** can fix the problem. Therefore, if **mount** fails, you should run **fsck.vfat** and try mounting again.

```
[root@wrap /]$ fsck.vfat -a device
```

Warning

There is not enough memory to run **fsck.vfat** on storage devices bigger than 8GB.

Note: If your application uses USB storage devices or Compact Flash memory cards for additional storage, you must ensure that these services do not start before these storage devices are properly mounted. You should therefore disable the automatic startup of application(s) in question either by changing their startup state to off in WWW Setup at Setup ⟶ Applications ⟶ Default startup applications or at shell prompt with command **chkconfig *application* off**. The system startup script should then be edited according to the following example for obexsender:

```
#!/bin/sh

# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
mkdir -p /mnt/disk
mount -t vfat /dev/sda1 /mnt/disk
if [ $? != 0 ]; then
    # run fsck.vfat only if mounting failed, then try to mount again
    fsck.vfat -a /dev/sda1
    mount -t vfat /dev/sda1 /mnt/disk
fi
service obexsender restart
```

3.7. Compact Flash GPS Card

The Access Server automatically identifies the Compact Flash GPS card when it is inserted. At that time, the device file `/dev/ttyS0` is created and the GPS card can be accessed by using that device with the serial port settings the GPS card uses.

The supported Compact Flash cards are listed in Appendix E.

3.8. Remote File Shares

3.8.1. Using NFS Mount

First, create a mountpoint: **mkdir -p /mnt/nfs**. To use the NFS mount, issue a command such as **mount -o nolock <ipaddr-of-server>:/sharename /mnt/nfs**. After this, you can access the share in directory `/mnt/nfs`.

When the share is not needed, unmount it with command **umount /mnt/nfs**

3.8.2. Using CIFS Mount

To use a CIFS mount (for example a shared folder in Windows), you need the `cifs-client` software component installed to Access Server or Access Point (see Section 3.2 for more information about installing software components). First, create a mountpoint: **mkdir -p /mnt/cifs**. Mount the directory by using command **mount.cifs //<ipaddr-of-server>/sharename /mnt/cifs -o user=username,nounix**. You will then be prompted for password of the username you specified. After entering the correct password, you can access the share in directory `/mnt/cifs`.

When the share is not needed, unmount it with command **umount /mnt/cifs**

3.8.3. Mounting at Boot Time

System startup script `rc.local` can be used to automatically mount remote file shares at bootup. Add the mountpoint creation and actual mount commands in that script. If your CIFS share needs a password, it can be added to mount options, for example **-o user=username,pass=password,nounix**.

3.9. Servers

Access Server and Access Point server applications are started automatically at system power-up or when an iWRAP server or the Internet services daemon needs them. You can check which servers are currently installed and/or configured to start at system power-up with command **chkconfig --list** or navigating in WWW Setup to Setup ⟶ Applications ⟶ Default startup applications. The servers and their purposes are described in Table 3-8.

Server	Description
bluetooth	Bluegiga iWRAP Server, which is described in Section 3.3 (details in Chapter 7).
connector	Bluegiga Connector, which automatically opens and maintains connections to specified Bluetooth devices. This server is configurable using the **setup** application and its WWW interface. See Section 4.1.4 for more information.
crond	A daemon to execute scheduled commands. This server is configurable through the `/var/spool/cron/crontabs/root` file or the **crontab** command in the same way as any Linux crond.

Server	Description
dhcpd	This server is a DHCP daemon for providing automatic network configuration for clients in the network. Notice that, by default, this server is only enabled for the `gn` interface, used by Bluetooth PAN Generic Networking profile. You can enable it for `nap` interface by using command **chkconfig dhcpd on** or from WWW Setup at Setup ⟶ Applications ⟶ Default startup applications. You will then need to configure static network settings at Setup ⟶ Network Settings ⟶ Default interface settings and ensure you have matching DHCP server settings in file `/etc/udhpcd-nap.conf`.
finder	WRAP Finder Service. See Section 3.9.1 for more information.
ftpd	Internet File Transfer Protocol Server. Notice that this server is not installed by default. You can install it from software component `ftpd` (see Section 3.2 for more information about installing software components). See Section 3.9.5 for more information about using FTP server.
httpd	Web server, which is described in detail in Section 3.9.6. Another Web server, `lighttpd`, is available as a separate software component. See Section 3.2 for more information about managing software components.
inetd	Internet services daemon. Notice that this server is disabled by default. Use the WWW interface of **setup** application or the **chkconfig inetd on** command to enable it. To configure inetd, edit its configuration file `/etc/inetd.conf`.
iptables	This is not a server, but a set of default IP packet filtering rules.
leds	This is not a server but a script controlling leds at power-up or system reboot.
maradns	A security-aware DNS server. Notice that this server is not installed by default. You can install it from software component `maradns` (see Section 3.2 for more information about installing software components). See http://www.maradns.org for more information about configuring and using maradns.
network	This is not a server but a networking control script.
nfsd	NFS server. Notice that this server is not installed by default. You can install it from software component `nfs-tools` (see Section 3.2 for more information about installing software components).
ntpd	Network Time Protocol (NTP) daemon. See Section 3.9.11 for more information.
obexsender	Bluegiga ObexSender server. See Section 3.9.2 for more information.
openvpn	OpenVPN™, a full-featured SSL VPN solution. Note that this server is not installed by default. You can install it from software component `openvpn` (see Section 3.2 for more information about installing software components). See Section 3.9.8 for more information about using OpenVPN™.

Server	Description
pppd	Point to Point Protocol daemon. GPRS network connections are established using **pppd**, and iWRAP server uses it with Lan Access Profile.
serialport	Bluegiga Bluetooth Serial Port Profile server. See Section 3.3.3 for more information.
smsgw	Bluegiga SMS gateway server. Notice that this server is not installed by default. Installation and usage instructions are described in detail in Section 3.4.3.
sshd	SSH daemon. See Section 3.9.9 for more information.
snmpd	SNMP daemon. Available in separate component `net-snmp`. See Section 3.9.7 for more information.
swap	This is not a server but script to start or stop swapping to NFS server.
syslogd	System logging daemon. This server can be configured by using the **setup** application or its WWW interface.
telnetd	Telnet protocol server. Notice that this server is disabled by default for security reason. Use the **setup** application or the **chkconfig telnetd on** command to enable it. See Section 3.9.10 for more information.
udhcpcd	DHCP client daemon for automatic network configuration.
watchdog	Bluegiga user level watchdog. See Section 3.9.3 for more information.
wpkgd	Bluegiga remote management system daemon. See Section 3.9.4
zcip	Zero configuration networking service.

Table 3-8. Access Server and Access Point Servers

3.9.1. Finder

The Finder service is a small service, which listens for UDP broadcast queries from Access Server and Access Point Finder applications and responds to those queries with identification information (IP address, model, serial number, etc.).

The **finder** command can be used to query Finder service information from Access Servers and Access Points in the network. With no parameters, **finder** sends the query using the broadcast address of the default interface (`nap`). Broadcasting to networks of other interfaces can be done with *`--interface`* parameter, such as the zero configuration interface `nap:9` in the following example:

```
[root@wrap root]$ finder --interface nap:9
Access Server 2291 (S/N: 0402110112) (build: 3.2)
- Description: Access Server #0402110112
- Hostname: wrap.localdomain
- IP: 10.1.1.51 (nap), 169.254.30.233 (nap:9), 192.168.161.1 (gn)
- Ethernet MAC: 00:07:80:00:03:ed
- iWRAP: 10101 00:07:80:80:0b:c3 bt2.0 (W0402110112_1)

Access Point 3201 (S/N: 0808270335) (build: 3.9cvs.1860)
- Description: Access Point #0808270335
- Hostname: wrap.localdomain
- IP: 10.1.1.60 (nap), 169.254.58.116 (nap:9), 192.168.161.1 (gn)
```

```
- Ethernet MAC: 00:07:80:01:16:c2
- iWRAP: 10101 00:07:80:89:58:2f bt2.0 (W0808270335_1)

[root@wrap root]$
```

With parameter *--send* **finder** will send info once to a specified host, for example to inform the host that the device has booted.

For information about the finder protocol, see Chapter 9.

3.9.2. ObexSender

The ObexSender application is automatically started in Access Server and Access Point. Its purpose is to receive business cards (vCards), images, or other files, and analyze their content and send files back selecting them based on configured keywords found.

ObexSender can also make an inquiry for Bluetooth devices, and automatically send one or more files to all new devices found.

ObexSender can be configured with the **setup** application or by editing the `/etc/obexsender.conf` file (see Section 2.4).

For detailed instructions on using ObexSender, see Chapter 5.

3.9.3. User Level Watchdog

Bluegiga User Level Watchdog daemon listens on UDP port 4266 for "id timeout" messages. "id" is an ASCII string, without spaces. If "timeout" equals to 0 (zero), the "id" is removed from the list of processes to wait. If "timeout" is greater than 0 (zero), the "id" is added or updated.

When there is no message for "id" received within the "timeout" seconds, the user level watchdog dies and the kernel watchdog reboots Access Server or Access Point.

The **watchdog** command can be used to send messages to the watchdog daemon. This is done through command **watchdog id timeout**. For example, **watchdog test 5**.

3.9.4. Remote Management

Access Server and Access Point contain simple tools that provide means for full and secure remote management of the device.

The basic remote management can be performed using the WWW Setup interface, SSH command line access, and SCP and SFTP file transfer protocols.

In addition to those, Access Server and Access Point contain Bluegiga Remote Management System for transferring management packets over different media to Access Server or Access Point and automatically sending response packets back.

The management packets (`*.wpk`) are automatically processed when they are transferred to the autoinstall directory in Access Server or Access Point (`/tmp/obex` by default, but configurable with the setup application or WWW interface at Setup ⟶ Applications ⟶ wpkgd settings). The easiest way to transfer a management packet to this directory is to upload it from WWW Setup at Setup ⟶ Advanced settings ⟶ Upload a software update.

3.9.4.1. Overview

A management action is performed using the following procedure:

1. A customer system prepares the management packet (*.wpk).
2. The management packet is delivered to Access Server or Access Point, to the packaging daemon's directory. You can currently use Bluetooth, SCP, SFTP and plain FTP to do this. The packet can also be transmitted using a USB memory dongle, Compact Flash memory card or through the WWW Setup interface.
3. The Access Server packaging daemon processes the management packet, possibly generating a reply packet.
4. (Optional) The reply packet is delivered to the customer system.

3.9.4.2. Management Packet Format

- The package name must be of format `name.architecture.wpk`, where "name" can be user defined and "architecture" is one of the supported architectures listed in Table 3-2.
- Package must be a `tar` archive that is compressed with **gzip** (such as files named *.tar.gz or *.tgz).
- The package must contain a package information file called `wpkg.pif` in the package root (the file contents are described later), otherwise the built-in defaults for `wpkg.pif` are used.
- All other files, if any exist, should be data files, scripts or executables required for the management operation.

3.9.4.3. Management Packet Information File Format

The management packet information file (`wpkg.pif`) consists of tags and their data, described below.

Note: Ensure that you do not have trailing white-spaces in tag data.

%wpkg-version: 2

Contains information for version checking. 2 is currently the only supported version. It is also the default value.

%wpkg-prepare: [command line[s]]

One or more commands (all commands are lines until the next tag is interpreted as a command line) to execute. Commands may contain parameters, redirections and job control as well.

The built-in default value for this is **/usr/bin/dpkg -i *.deb || echo ERROR: Installation failed.**. This enables the special case of creating .wpk packets from .deb packets simply with **tar czf foo.wpk foo.deb**. (`wpkg.pif` is not needed in this special case).

%wpkg-reply: method

This value indicates where the generated reply packet is sent. By default, it is sent to where it came from. Possible values are:

- default

- file:///path/filename
- scp://remote:file
- objp://bdaddr/
- none

%wpkg-format: type

This value indicates what kind of a reply packet will be generated. Possible values are:

- ascii (this is the default value, everything echoed by the prepare-section will be sent).
- tgz (all files in the current directory will be sent).
- vcf (same as ascii, but assume it is a vCard).
- vmg (same as ascii, but assume it is a vMessage).
- vnt (same as ascii, but assume it is a vNote).
- vcs (same as ascii, but assume it is a vCalendar).
- html (same as ascii, but assume it is HTML).

%wpkg-option: value

Optional additional parameters. Supported values are **dupe**, **dupe1h** and **dupe1d**. These can be used to avoid installation of the same packet twice from a USB dongle (once when inserted and again after reboot, if dongle is still inserted when Access Server or Access Point reboots next time). Packaging daemon does this by inserting a serial number and a timestamp to a file `packet_name.dupe`. When the processing of the management packet is about to start again, packaging daemon checks the file and aborts the installation if a timestamp is found (in the case of the **dupe** parameter) or if the timestamp is less than an hour (in the case of **dupe1h**) or less than a day (in the case of **dupe1d**) old.

%wpkg-auth: auth

Optional authentication string required by wpkgd.

3.9.4.4. Management Operation Example: Hello World

See below for the simplest example of `wpkg.pif`:

```
%wpkg-version: 2
%wpkg-prepare:
echo Hello world
```

This will generate a reply packet containing text "Hello world". You can generate the wpk file simply by giving the command **tar czf hello.noarch.wpk wpkg.pif**.

3.9.4.5. Management Operation Example: Software Update

See below for a more complex example of **wpkg.pif**:

```
%wpkg-version: 2
%wpkg-prepare:
FOO=`pwd`
```

```
cd /
tar xzf ${FOO}/files.tar.gz
echo Done.
```

This example will extract files from the included `files.tar.gz` file. You can generate the wpk file with command **tar czf update.arch.wpk wpkg.pif files.tar.gz**.

Note: See Table 3-2 for possible values of "arch" in WPK filename. If you need to install applications on all supported platforms, you must use the Bluegiga SDK to build binaries to all architectures and provide separate installation packets to all of them; alternatively, your `wpkg.pif` can analyze which architecture is supported with command **wrapid --hw** and extract correct binaries.

3.9.4.6. Management Operation Example: IPQUERY

In this example, we build a simple packet that can be used with a Bluetooth enabled phone to retrieve the IP address of an Access Server or Access Point. File `wpkg.pif` reads:

```
%wpkg-version: 2
%wpkg-format:  vcf
%wpkg-prepare:

ipaddr() {
echo `ifconfig nap | grep "inet addr" | awk -F [:] \
  \\{print\\$2\\} | awk \\{print\\$1\\}`
}

serialno() {
echo `wrapid | grep Hardware | awk \\{print\\$5\\}`
}

echo -e "BEGIN:VCARD\r"
echo -e "VERSION:2.1\r"
echo -e "N:`serialno`\r"
echo -e "TEL:`ipaddr`\r"
echo -e "URL:`hostname`\r"
echo -e "END:VCARD\r"
```

This example will send the reply back as a vCard (contact card). Please note that you have to include all required vCard formatting by yourself. You can generate the wpk file simply giving the command **tar czf ipquery.noarch.wpk wpkg.pif**.

To use this example, send the file `ipquery.noarch.wpk` to the inbox of your Bluetooth phone. Check that you have Bluetooth enabled in the phone. Then, from the phone's inbox, send the file `ipquery.noarch.wpk` over Bluetooth to Access Server or Access Point.

3.9.4.7. Management Operation Example: Beep

See below for beep example (for Access Server only) of `wpkg.pif`:

```
%wpkg-version: 2
%wpkg-reply:   none
%wpkg-prepare:
echo A > /dev/led
sleep 1
```

```
echo a > /dev/led
```

3.9.4.8. Management with USB Memory Dongle or Compact Flash Memory Card

When a USB memory dongle or Compact Flash memory card is inserted, Access Server and Access Point automatically try to mount it using the VFAT filesystem. If the mount is successful, the root directory is scanned for `*.wpk` files. If one is found, the Bluegiga Remote Management System daemon processes it. Optional reply packets are saved back to the root directory (unless otherwise stated in the **%wpkg-reply** tag). It is good practise to prevent accidental duplicate processing of management packets with **%wpkg-option** tag.

3.9.5. FTP

If you install the FTP server (from software component package `ftpd)`, users can use it to log in anonymously to the `/tmp/obex` directory with download access or as **`root`** with password **`buffy`** to the root directory with full access. The password and other settings can be changed on Access Server and Access Point with the **setup** application or by editing the `/etc/ftpd.conf` file (see Section 2.4).

> **Note:** Do not use FTP because it is insecure. Use SSH (SCP or SFTP) instead. A commonly used client with a graphical user interface is, for example, WinSCP (http://winscp.net/).

3.9.6. Web Server

The integrated web server in Access Server and Access Point support HTTP/1.0 methods GET and POST, and has light user authentication capabilities. The content can be either static or dynamic - the WWW server is CGI/1.1 compatible.

The web server is always running and the content (http://wrap-ip-address/) is located in the `/var/www/html/` directory in Access Server's and Access Point's file system.

The web server is configured to protect the WWW Setup interface with a username and password. The default username and password can be changed as instructed in Section 2.4. For further information about using the web server for your own applications, see the web examples in Section 6.3.1.

3.9.7. SNMP

You can install the Net-SNMP suite of applications to Access Server or Access Point from software component `net-snmp` (see Section 3.2 for more information about installing software components). The current Net-SNMP implementation for Access Server and Access Point is limited. However, it can be used to poll the basic status of Access Server or Access Point.

Configuration details can be found and altered in configuration file `/etc/snmp/snmpd.conf`, which is accessible as described in Section 2.4.

For more information about the Net-SNMP suite, see http://net-snmp.sourceforge.net/

3.9.8. OpenVPN

You can install OpenVPN™, a full-featured SSL VPN solution, to Access Server or Access Point by installing software component `openvpn` (see Section 3.2 for more information about installing

software components).

For detailed instructions on using OpenVPN with Access Server and Access Point, see Section 10.4.

For more information about the OpenVPN™, see http://openvpn.net/.

3.9.9. SSH

By default, users can use SSH to log in (or SCP and SFTP to transfer files) as user **root** with password **buffy**. The password can be changed on Access Server and Access Point by using command **passwd** or with the **setup** application.

3.9.10. Telnet

If you enable telnet, users can log in over telnet as user **root** with password **buffy**. The password can be changed on Access Server and Access Point using the command **passwd** or with the **setup** application.

> **Note:** Do not enable telnet because it is insecure. Use SSH instead.

3.9.11. NTP

The **ntpd** service uses the standard Network Time Protocol (NTP) to keep Access Server and Access Point system time automatically in sync using a random selection of eight public stratum 2 (NTP secondary) time servers. You can configure the NTP server to retrieve the correct time from a single time server by using the setup application or its WWW interface, at Setup ⟶ Network Settings ⟶ Time server (NTP). The service is also configured to answer NTP requests from other devices.

The NTP server configuration can also be altered by editing its configuration file `/etc/ntpd.conf`.

> **Tip:** Access Server and Access Point can provide RFC 868 time service with inetd daemon. You need to enable inetd daemon (see Section 3.9 and enable the time service by editing its configuration file `/etc/inetd.conf`

3.10. Utilities

Access Server and Access Point are basically small Linux systems. Whether logged in from the management console or with SSH, your shell session starts as the root user in the root directory. After that, you have the option to use most of the standard Linux utilities. For detailed list of available commands, see Appendix C and Appendix D.

3.11. Real Time Clock

The system clock is read from the battery operated real time clock during boot. The system time is automatically written to the real time clock when the system is rebooted using the **reboot** command. This can also be done using the **hwclock --systohc --utc** command. Give command **hwclock --help** for more information about the **hwclock** utility.

Tip: Easiest way to set correct time is to use setup application or its WWW interface by navigating to Setup ⟶ Network settings ⟶ Update current time now by NTP. It will also save the time to the battery operated real time clock.

3.12. Time Zone

The default time zone in Access Server and Access Point is UTC. You can change it by installing correct `tzdata*wpk` management packet, available from http://update.bluegiga.com/as/4.0/timezone/ or Bluegiga Software Development Kit DVD-ROM.

3.13. System Information (wrapid)

You can get detailed information of Access Server and Access Point hardware and software with setup application or its WWW interface by navigating to Setup ⟶ Advanced settings ⟶ Hardware information. At shell prompt, running command **wrapid** outputs the same information. If you need to use a hardware information detail for example in your own shell scripts, you can ask it directly with **wrapid** command. Run it with parameter `--help` for list of queries it supports (hardware serial number, software version number and so on).

3.14. Software Upgrade

Access Server and Access Point can be upgraded to the latest software version. The latest software updates and instructions are available at http://techforum.bluegiga.com/.

Note: Upgrading to software version 4.0 is only possible with a reflash package, which will erase all existing information, reset all passwords to their defaults and regenerate SSH keys.
If you have your own applications running in Access Server or Access Point, stop and uninstall them first.

The easiest way to install the latest software version is to do it with a USB memory dongle:

1. Find the correct software upgrade packet for your Access Server or Access Point architecture (see Table 3-2 for information of architectures) and copy the correct `reflash-[version].[architecture].wpk` file to an empty USB memory dongle.
2. Power down Access Server or Access Point.
3. Insert the dongle in Access Server or Access Point.
4. Power up Access Server or Access Point.
5. Wait with the dongle inserted for the blue leds to start blinking from side to side.
   Note: Do not power down Access Server or Access Point while blue leds are blinking from side to side or all turned on.

   Installation takes 5-15 minutes, be patient.
6. Check that only led labeled "1" in Access Point (blue led closest to the power led in Access Server) turns on and off every 4 seconds. You will then also see Bluetooth led, (led labeled "2" in Access Point and blue led furthest away from power led in Access Server) to blink quickly every 30 seconds indicating Bluetooth service activity.
7. You have now successfully upgraded Access Server or Access Point.

Note: If you hear beeps (in case of Access Server) and all blue leds start blinking on and off at the same time, you have tried to upgrade with a wrong `reflash-[version].[architecture].wpk` packet. You can confirm this from a log file in the root directory of your USB dongle. The log file is a `txt` file named using the software upgrade packet's filename and system timestamp. Please check again which file you should have used with help from Table 3-2 and try again.
Note: In some rare occasions the update process of an old Access Server may hang. If after 15 minutes all blue leds are still on, please power down Access Server, remove duplicate install protection file called `reflash-[version].[architecture].wpk.dupe` from USB dongle and restart the installation process.
Note: Instead of using a USB dongle, you can install software upgrade by uploading the packet with WWW Setup at Setup ⟶ Advanced settings ⟶ Upload a software update. If you need to find the installation error log, it can be found by browsing to directory `/dev/shm/tmp/obex` at Setup ⟶ Advanced settings ⟶ Browse all files.

3.15. Factory Reset

Access Server and Access Point can be reprogrammed with the latest software version, erasing all data and recovering a system that does not boot up normally.

Note: The latest software updates and instructions are available at http://techforum.bluegiga.com/.

The easiest way to install the latest software version is to do it with a USB memory dongle:

1. Copy `kernel.*` and `root.*` files to an empty USB memory dongle. You can find these files inside `factoryreset.zip` available at http://techforum.bluegiga.com/, or in directory `dev/shm/phantom` inside `reflash*.wpk` packets (just rename the packets to `*.tgz` files and unpack with for example `tar` or WinZip)
2. Insert the dongle in Access Server or Access Point.
3. Power up Access Server or Access Point.
4. Wait with the dongle inserted as long as all blue leds are on. You will need to wait for 5 minutes when reprogramming Access Point and 10 minutes when reprogramming Access Server.
5. Check that only led labeled "1" in Access Point (blue led closest to the power led in Access Server) turns on and off every 4 seconds. You will then also see Bluetooth led, (led labeled "2" in Access Point and blue led furthest away from power led in Access Server) to blink quickly every 30 seconds indicating Bluetooth service activity.
6. You have now successfully reprogrammed Access Server or Access Point.

Chapter 4. SPP-over-IP

SPP-over-IP is a special functionality of iWRAP Bluetooth servers running in Access Servers and Access Points. It offers a transparent way to transmit data from Bluetooth Serial Port Profile (SPP) enabled devices to server computers or PCs. Several transport medium are supported, such as ethernet, Wi-Fi and GPRS.

4.1. How SPP-over-IP Works

The SPP-over-IP application enables transparent data transfer between any Bluetooth Serial Port Profile (SPP) compliant device and a server, laptop or desktop connected to the same network. This enables plug n' play connectivity from a Bluetooth network to any standard TCP/IP based network. See Figure 4-1 for an overview of the application and a brief introduction to its functionality.

Features of SPP-over-IP are:

- Access Server 2291 and Access Point 3201 support 7 incoming SPP connections.
- Access Server 2292 supports 14 incoming SPP connections.
- Access Server 2293 supports 21 incoming SPP connections.
- SPP-over-IP can be used over ethernet, Wi-Fi or GRPS networks.
- SPP-over-IP also works over Bluetooth Personal Area Networking (PAN) connections, so not all Access Servers or Access Points need to be physically (cable) connected to the TCP/IP network, but some Access Servers or Access Points can be linked using the Bluetooth PAN connection. This is referred to as *repeater* operation.
- If SPP-over-IP application cannot open the TCP connection to defined IP address and port, the SPP connection will not be accepted.
- If the TCP server on PC is closed, all SPP connections will be closed as well.
- When Access Server or Access Point is in its default configuration, it tries to enable sniff power saving mode on all idle Bluetooth connections to minimize power consumption.
- SPP-over-IP can also be used to opposite direction, i.e. Access Server or Access Point opens the Bluetooth connections to dedicated Bluetooth devices. See Section 4.1.4 for more details.
- SPP-over-IP can also be combined with the Tactical Software's Serial/IP® software. Serial/IP software converts automatically TCP connections to virtual COM ports on the host PC, so legacy applications utilizing COM-ports instead of TCP/IP can also be used.

4.1.1. Standard Operation

With the standard configuration, SPP-over-IP works as described below:

- Listens for incoming Serial Port Profile (SPP) connections
- Takes control of all incoming connections
- Opens a TCP connection to the defined IP address and TCP port

- Forwards all incoming data from the SPP device to the established TCP connection and vice versa

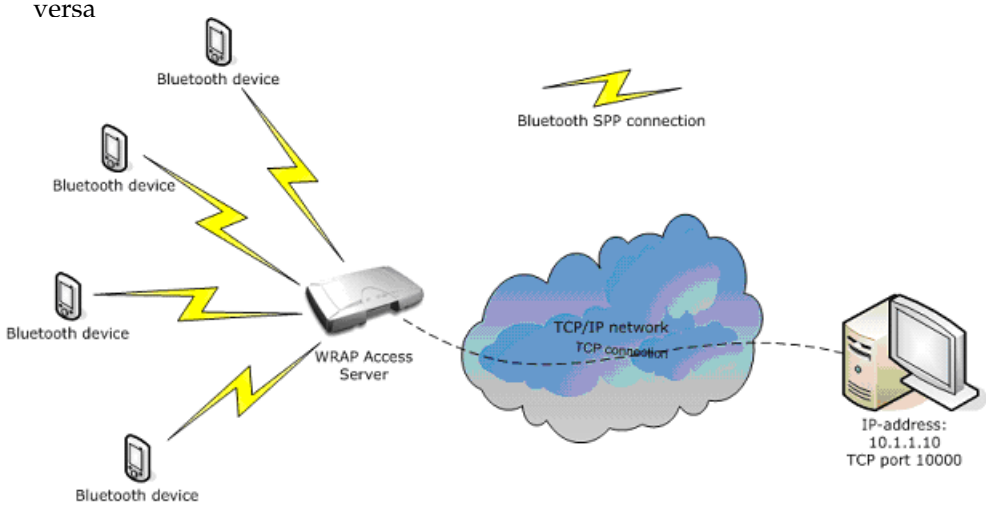

Figure 4-1. SPP-over-IP Network Architecture

All the server computer needs to do is to listen for incoming TCP connections from Access Server or Access Point to a specified TCP port and receive/send the application data.

4.1.2. Repeater Operation

The SPP-over-IP application can also be used in what is known as repeater mode. This feature is useful when all Access Servers or Access Points can not be directly connected to the TCP/IP network, but they can be connected to other Access Servers or Access Points by using Bluetooth PAN-connection. PAN enables transmitting TCP/IP packets wirelessly over Bluetooth. The figure below illustrates this configuration:

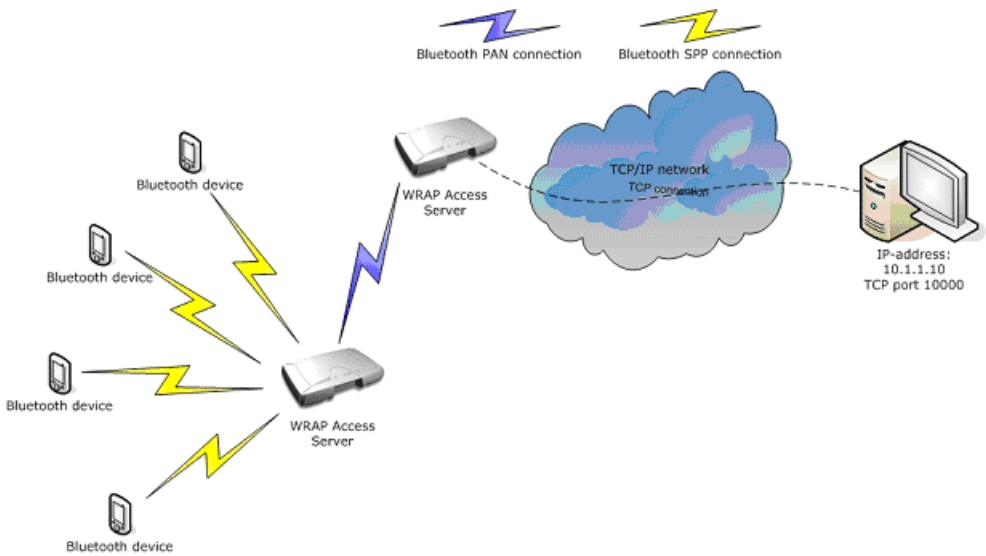

Figure 4-2. Repeater Mode in SPP-over-IP

4.1.3. SPP-over-IP over GPRS

SPP-over-IP software can also be used over GPRS instead of wired ethernet connection. This requires that Access Server or Access Point is equipped with a working GSM/GPRS compact flash or USB modem. See Appendix E for supported cards.

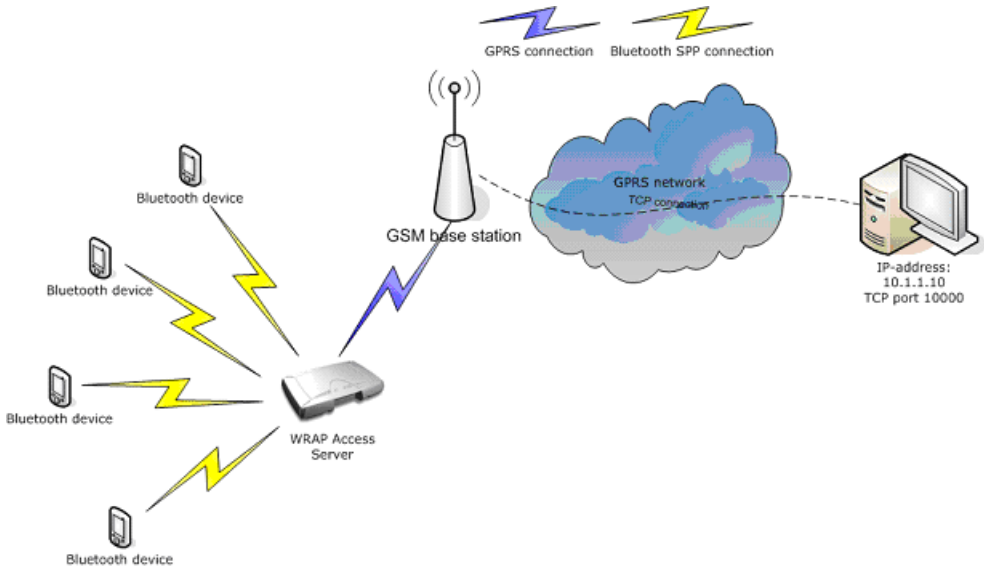

Figure 4-3. SPP-over-IP over GPRS

Notice when using GPRS:

- Data upload rate is around 8-12kbps (depending on GPRS card)
- Data download rate is around 32-48kbps (depending on GPRS card)
- Data transmission delays can be very high, sometimes even seconds
- GPRS connection may be unreliable and break easily. This should be taken account when designing the system. If the GPRS connection breaks, all the TCP and Bluetooth connections will also be closed.

4.1.4. Opening Connections from Access Server

In the basic SPP-over-IP use case, Access Server and Access Point are in passive mode and only accept incoming connections. Using the **connector** service, Access Server or Access Point can open and maintain outgoing Bluetooth connections to defined Bluetooth devices.

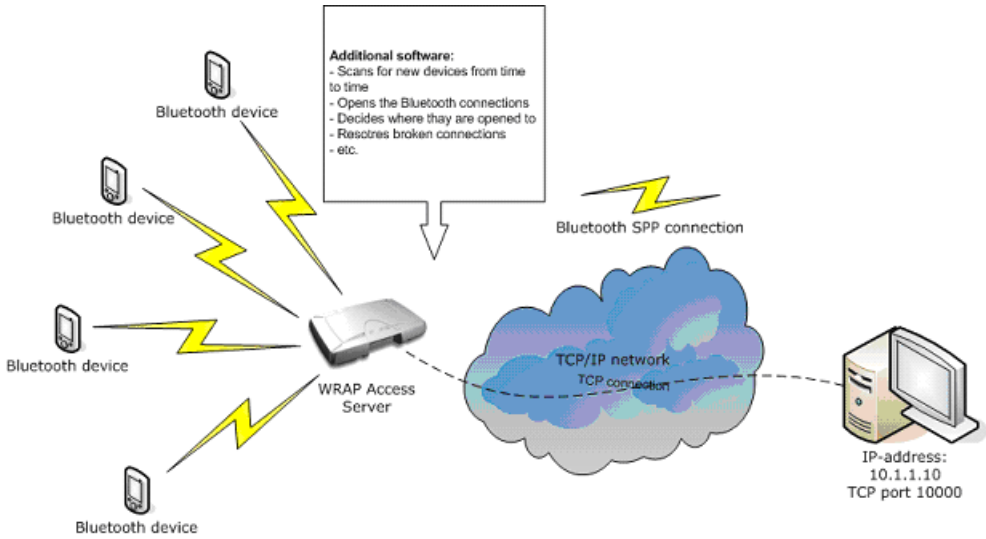

Figure 4-4. Access Server Opening the Connections

4.1.5. SPP-over-IP and COM Ports

SPP-over-IP can also be used together with Tactical Software's Serial/IP® software. Serial/IP software simply converts the TCP connections into virtual COM ports on the host computer. This is very useful in applications, which do not have support for TCP/IP but support COM ports instead.

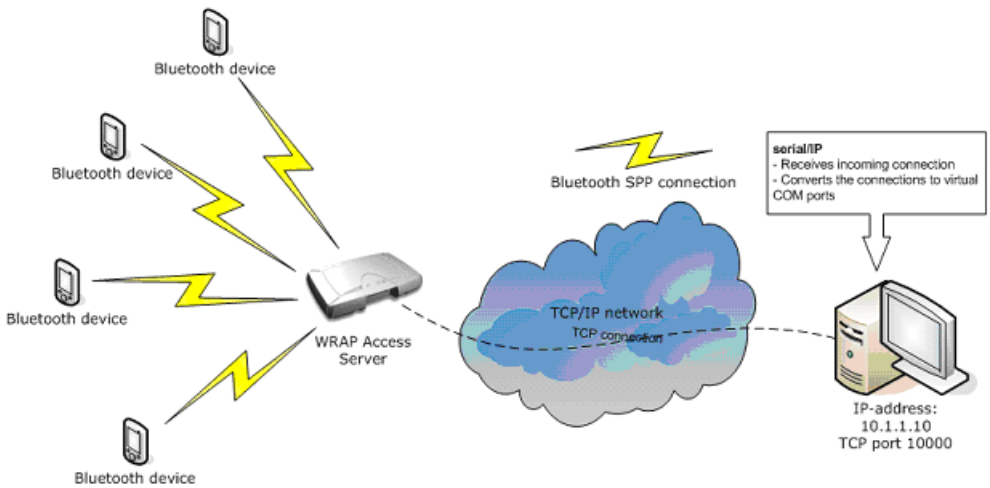

Figure 4-5. SPP-over-IP with Serial/IP

An evaluation version of Serial/IP can be downloaded from: http://www.tacticalsoftware.com/products/serialip.htm

4.2. Configuring SPP-over-IP

This chapter briefly instructs you to configure SPP-over-IP to work in different network setups or use cases.

SPP-over-IP is easiest to configure through WWW setup, which allows you to access all the necessary configurations. For instructions about finding Access Server's or Access Point's IP address and using the WWW setup interface, see Section 2.2.

4.2.1. Forwarding Incoming Connections

The basic SPP-over-IP operation, listening incoming Bluetooth connections and forwarding them to a TCP/IP socket on a remote host (or a local application), is configured at Setup ⟶ iWRAP settings ⟶ Bluetooth profiles ⟶ Connection forwarding. For details of the settings, see Section B.5.1.5

4.2.2. Maintaining and Forwarding Outgoing Connections

The SPP-over-IP **connector**, which opens and maintains outgoing Bluetooth connections and forwards them to a TCP/IP socket on a remote host (or a local application), is configured at Setup ⟶ Applications ⟶ Connector. For details of the settings, see Section B.4.1

4.2.3. Repeater Configuration

If you want to configure Access Server or Access Point to also act as a repeater (see Figure 4-2) you must make some additional configurations. Add the line below to your Bluetooth startup script, editable at Setup ⟶ iWRAP settings ⟶ Edit startup script. The line starting with # is a comment, which can be left out:

```
# Automatically connect to Access Server with PAN-NAP enabled using baseband 1
10101 SET CONTROL AUTOEXEC CALL 00:07:80:80:bf:01 PAN-NAP
```

You must replace the Bluetooth address used in the example (00:07:80:80:bf:01) with the Bluetooth address of the Access Server or Access Point, on which you want to receive the PAN connection.

> **Note:** The server receiving the PAN connection must have the PAN-NAP profile enabled. This is by default not the case, so in setup or its WWW interface, ensure that the setting at ⟶ Bluetooth settings ⟶ Bluetooth profiles ⟶ Enable PAN network access point profile says yes. No other configuration is needed. See Section 3.3.5 for more information on PAN profiles.

The Bluetooth PIN codes must be the same in both devices.

In the example configuration (Figure 4-6) connection forwarding has already been configured using the WWW Setup (two lines above the **SET CONTROL AUTOEXEC** line).

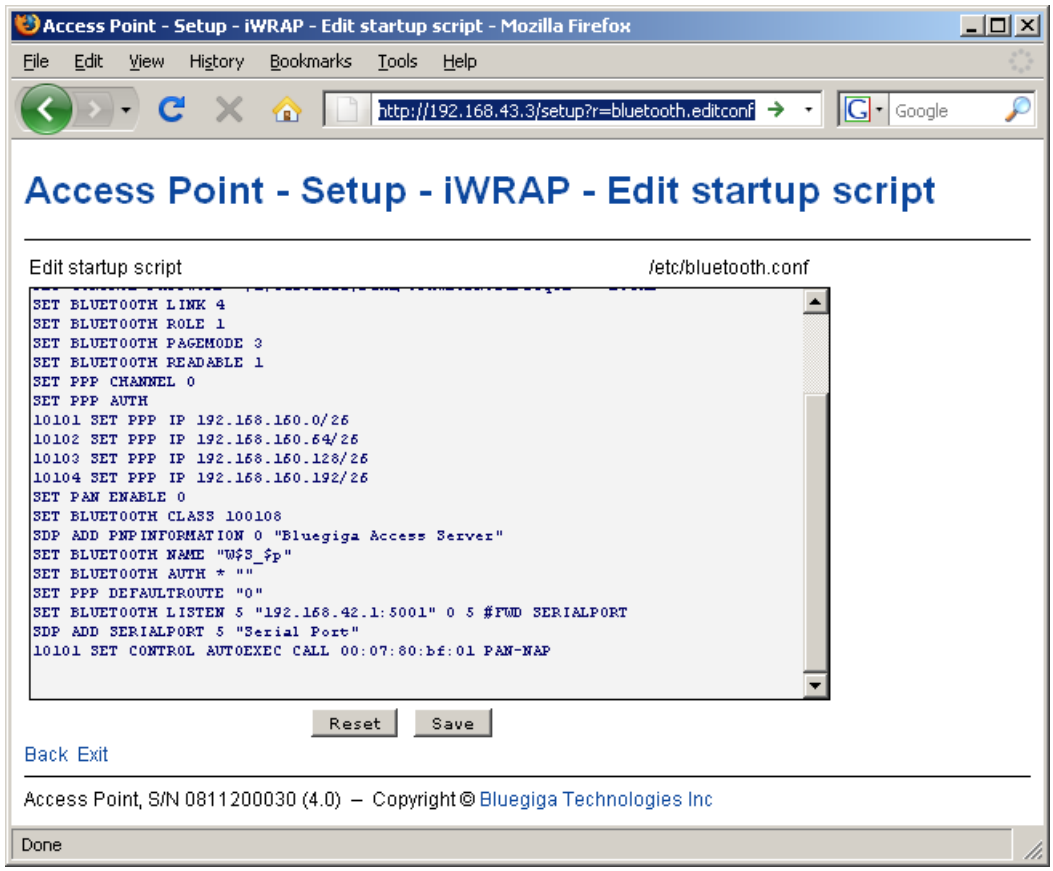

Figure 4-6. Repeater Configuration

4.2.4. Wi-Fi Configuration

If Access Servers or Access Points must be connected to Wi-FI (WLAN) instead of physical ethernet connection, you also need to make additional configurations through the WWW setup.

See Section 3.5 for more information.

4.2.5. GPRS Configuration

If Access Servers or Access Points must be connected to GPRS network instead of physical ethernet or Wi-Fi connection, you also need to make additional configurations through the WWW setup.

See Section 3.4.2 for more information.

Chapter 5. Obexsender

Obexsender is one of the built-in applications in Access Server and Access Point. It is dedicated to Bluetooth proximity marketing, content distribution, location based services, and much more. Access Server or Access Point with Obexsender provide the user with a ready platform to start content distribution including all the necessary Bluetooth functions from discovering the devices to transmitting the content. The user only needs to focus on what, when, and to whom to send the content - rest is taken care of by Obexsender.

The figure below illustrates a simplified Obexsender network:

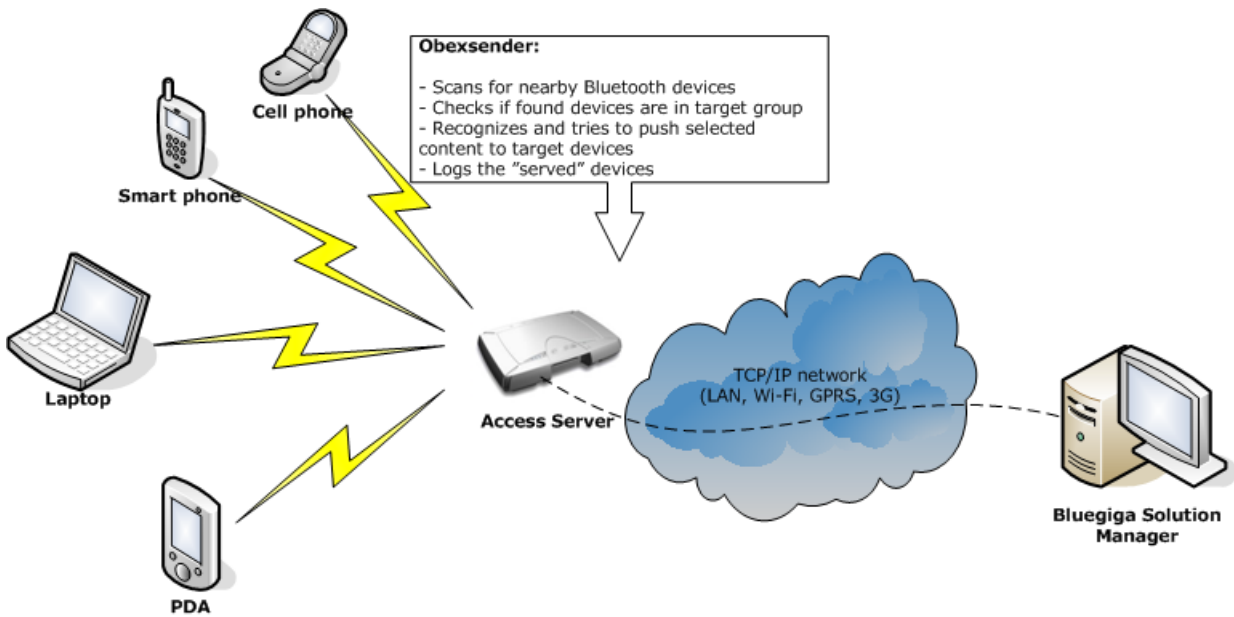

Figure 5-1. Simplified Obexsender network

5.1. Key Features

- Automatic device discovery and content push over a Bluetooth connection
- 18 simultaneous Bluetooth connections with one Access Server 2293, 12 with Access Server 2292, 6 with Access Server 2291 or Access Point 3201
- Upload speed even up to 75KB/sec with Bluetooth 2.0+EDR
- Content can be stored locally - with external memory even gigabytes of storage can be used
- Wide networking support: Bluetooth, ethernet, Wi-Fi, GPRS and EDGE
- Secure remote connections over a Virtual Private Networking
- Remote file system support
- Lots of filtering options, such as device type, or distance from Access Server or Access Point
- Extensive logging

- Interaction between several Access Servers or Access Points
- Content time stamping

5.2. Use Cases

This chapter describes some possible ObexSender use cases.

5.2.1. Content Push

This is the standard functionality in ObexSender. In the content push mode, ObexSender scans for devices and pushes content to clients who belong to the target group (not opted out by filtering).

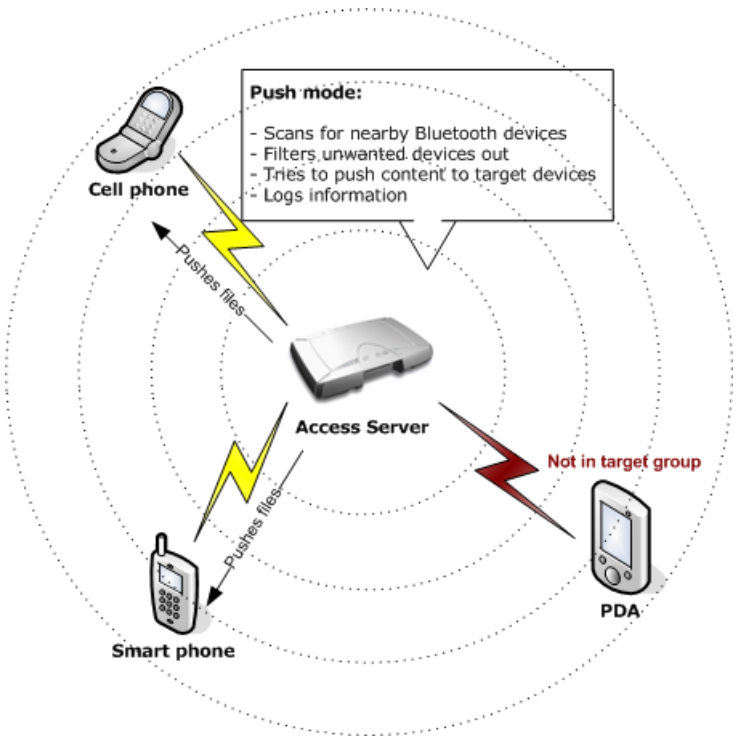

Figure 5-2. ObexSender Use Case: Content Push

5.2.2. Content Pull

ObexSender can also be configured into a content pull mode. In this mode, the transaction is initiated by the user. The user can send any file to the server or alternatively a file containing some specific string such as "MP3" or "NOKIA N73". The server parses the received file and as a response pushes a corresponding file to the user if such exists.

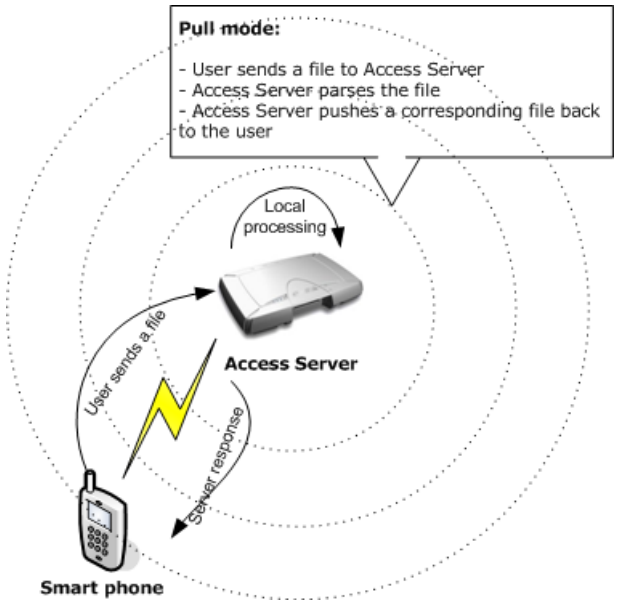

Figure 5-3. ObexSender Use Case: Content Pull

5.3. Configuration

This chapter contains basic ObexSender configuration instructions. The easiest and fastest way to configure ObexSender is through the WWW setup. For instructions about finding Access Server's or Access Point's IP address and using the WWW setup interface, see Section 2.2.

For details and default values of ObexSender configuration, please study the help texts in WWW Setup interface (also in Section B.4.2) and the ObexSender configuration file, which can be viewed and edited at Setup ⟶ Applications ⟶ ObexSender settings ⟶ Edit configuration file.

> **Note:** ObexSender will exit at startup by default, as it is not configured to do anything (there are no active send or reply rules in the default configuration).

5.3.1. Uploading Files

ObexSender needs content (files) to be sent for users. These can easily be uploaded by navigating to Setup ⟶ Applications ⟶ ObexSender settings ⟶ Upload a new file. All you need to do is browse for the file you want to upload and click Upload. You will see a confirmation note, for example *"File /usr/local/obexsender/files/pic.jpg uploaded"* .

You can also use secure FTP to transfer files (normal FTP is not installed by default in Access Server or Access Point for security reasons). For example WinSCP, available from http://www.winscp.org/, is a good application for secure FTP file transmissions.

5.3.2. Configuring Content Rules

Specifying the content (files) to be sent by ObexSender is done by adding **send** and **reply** directives to ObexSender configuration file `/etc/obexsender.conf`. The file is editable at Setup ⟶ Applications ⟶ ObexSender settings ⟶ Edit configuration file and also contains usage examples of both directives.

5.3.3. How to Store Files Sent to Access Server or Access Point

By default, all files sent over Object Push to Access Server or Access Point are stored to the `/tmp/obex` directory and deleted after they have been processed. It is however possible to save a copy of the file to another local directory or even to a remote location before it is deleted. This is configured by adding the *`--fork`* parameter running extra command before deletion to the Optional parameters for server setting in Setup ⟶ iWRAP Settings ⟶ Bluetooth profiles ⟶ Object push profile settings menu.

Example parameters storing a copy of file:

To a local directory

You can save a copy of the file received to a local directory using **cp** command like in the following example:

```
--bdaddr $b --prefix $b-$P- --fork '/bin/cp $$t /tmp/$$p$$d-$$T'
```

Now, after reboot, all incoming files are copied to `/tmp` directory. The format of the files is `bdaddr-btserverport-timestamp-filename.ext`.

To a remote location securely using SSH

You can save a copy of the file received to a remote location using **scp** command like in the following example:

```
--bdaddr $b --prefix $b-$P- \
--fork '/usr/bin/scp $$t remoteuser@remote.server:target_path/$$p$$d-$$T'
```

Now, after reboot, all incoming files are copied to remote server using SSH.

Note: You must enable **`root`** user of the Access Server or Access Point executing the **scp** command to log in as **`remoteuser@remote.server.address`** without password query (using public keys). See your SSH documentation for more information on how to do this. Under Windows, this is possible by using for example OpenSSH for Windows.

To a remote WWW server

You can install the `curl` software component (see Section 3.2 for more information on installing software component) which enables you for example to post a copy of the file to a WWW form, like in the following example:

```
--bdaddr $b --prefix $b-$P- \
--fork '/usr/bin/curl -F upload=@$$t http://remote.server/remoteform.php
```

For more information of cURL, see http://curl.haxx.se/docs/manpage.html

To a remote location using other protocols

Generally, if you have an application that can send a file to a remote location using some protocol, it can be used to transfer files sent to Access Server or Access Point. In addition to above, for example FTP and SMTP are supported with Bluegiga software. Visit for more information. (http://techforum.bluegiga.com/)

5.4. Monitoring ObexSender

ObexSender logs its operation using syslog or to a specified log file (configurable via WWW setup).

When you choose View log in the ObexSender menu, you can only see the summary of ObexSender action, that is how many successes, failures and retries have occurred. When you select the date or Total in the summary view, you will see more details. You will see to which Bluetooth address the content was sent and if the transmission was a failure or success, or if transmission will be retried later.

5.5. Bluetooth Device Database

ObexSender uses a special Bluetooth device database for recognizing Bluetooth devices. If you find devices that are not identified properly, please send the output of command Setup ⟶ Applications ⟶ ObexSender settings ⟶ Inquiry and calculate hash to <support@bluegiga.com> together with detailed information, or at least the exact model of device identified with its Bluetooth address and friendly name, of devices found with that command, but improperly identified by ObexSender. Bluegiga Technologies will then provide you with the latest Bluetooth device database.

Chapter 6. Software Development Kit

6.1. Introduction to SDK

This chapter describes how to create and use applications by using Bluegiga Software Development Kit.

The software running in Access Server and Access Point can be divided to following categories:

1. Linux kernel and boot loader.
2. Bluegiga kernel device drivers (led+io, bbreset).
3. Bluegiga iWRAP Bluetooth stack, profiles (SPP, PAN, ObjP, FTP) and applications (for example ObexSender, connector and btcli).
4. Bluegiga servers (for example finder, smsgw and wpkgd) and applications (for example setup and chkconfig).
5. GPLed applications (for example BusyBox, OpenSSH and bash).
6. Applications written with Software Development Kit.

See Appendix C for software details.

6.2. Installing SDK

The following hardware and software are required to run the Bluegiga Software Development Kit:

A PC with:

- Supported i386 Linux distribution with required software packages installed.

  SDK has been tested with Ubuntu 8.10, Ubuntu 8.04, CentOS 5.2, Fedora 10, Fedora 9, Foobar Linux 5, RedHat Enterprise Linux 5.3 and openSUSE 11.1. SDK has also been tested on x86_64 platform.

  If you are going to install the pre-compiled toolchain to its default location `/usr/local/arm`, basic compiler tools and only `zlib` and `bzip2` development libraries (and `gettext` package in Ubuntu and `patch` in openSUSE) are needed to be installed. In Fedora, CentOS and RHEL5, this can be done with command **sudo yum install gcc make zlib-devel bzip2-devel**, in Ubuntu with command **sudo apt-get install build-essential zlib1g-dev libbz2-dev gettext** and in openSUSE with command **sudo zypper install gcc make zlib-devel libbz2-devel patch**.

  If you want to install the toolchain to a custom location, it needs to be recompiled, which can take hours to complete. You will also need to install additional packages. In Fedora, CentOS and RHEL5, these can be done with command **sudo yum install gawk bison flex libtool automake texinfo ncurses-devel**, in Ubuntu with command **sudo apt-get install gawk bison flex libtool automake texinfo libncurses-dev** and in openSUSE with **sudo zypper install gawk bison flex libtool automake texinfo ncurses-devel**.

- 2GB of available hard disk space (4GB if you want to install the toolchain to a custom location)

- An ethernet connection to a Local Area Network (also connected to Access Server or Access Point) is highly recommended. If you are installing the toolchain to a custom location, connection to Internet is required as toolchain source code needs to be downloaded.

To install the SDK, mount the Bluegiga SDK DVD-ROM or ISO image, change the current working directory to where it is mounted, and run the **sdk-install** script.

> **Note:** The user running installing the SDK needs to have privileges to do the mounting and to create the directory for the toolchain (by default in directory `/usr/local/arm`).

If you are installing the toolchain to default location, you can optionally install toolchain sources too. The **sdk-install** script will ask if you want to do this.

If you want to install the toolchain to a custom location, the **sdk-install** script will automatically choose to install toolchain sources and try to recompile the toolchain.

Example installation (user input is printed **like this**):

```
$ mount /dev/cdrom /mnt/cdrom
$ (or mount -o loop /path/to/sdk.iso /mnt/cdrom)
$ cd /mnt/cdrom
$ sh sdk-install

Bluegiga SDK installation

NOTE! If you change toolchain directory entire toolchain must be recompiled.
Recompile process is very complicated and may fail on your system.

Toolchain directory: [/usr/local/arm] ENTER
SDK directory: [/home/user/asdk] ENTER
Install toolchain sources? [y/N] ENTER

Starting installation in 5 seconds... Please wait or press Ctrl-C to abort.

Installing toolchain binaries (98M)... done
Installing SDK (213M)... done
Compiling test image...

### compiling output deleted ###

Image compiled successfully to "/home/user/asdk/tmp/"

Installation done.
$
```

Toolchain directory is the path where you want the Bluegiga Software Development tools (`arm-linux-gcc`, etc.) to be installed.

SDK directory is the path where you want the Bluegiga Software Development Kit to be installed.

6.3. Creating Applications

The fastest way to start developing Access Server applications is to study, change, and recompile the example files in the `asdk/examples` directory.

6.3.1. Application Examples

To demonstrate the software development features of Access Server or Access Point, the Bluegiga Software Development Kit comes with several example applications.

6.3.1.1. Installing Examples

The compiled example files are located in WPK packets on the Bluegiga SDK tree in subdirectories `asdk/examples/{name_of_example}/_{architecture}`. (see Table 3-2 for description of supported architectures).

The examples can be manually uploaded and installed on Access Server or Access Point by sending the WPK packet to the `/tmp/obex` directory. The **wpkgd** server automatically installs them. Uploading can be done over Bluetooth, SCP, SFTP or WWW Setup ⟶ Advanced settings ⟶ Upload a software update (see Figure 2-16).

If your Access Server or Access Point is connected to Internet, you can also install the examples from the Bluegiga software update repository with command line with command: **wpkgd install name_of_example**.

6.3.1.2. Running Examples

The examples, with their usage and purpose, are described in Table 6-1.

Example	Usage	Purpose
helloworld	**/usr/bin/helloworld**	The "Hello, world!" application.
serial	**/usr/bin/serial /dev/ttyAT1**	"Hello, world!" to the serial port. Notice that `/dev/ttyAT1` must be free (no Bluegiga SMS Gateway or Bluetooth Serial Port Profile is using it).
forkserver	**SET BLUETOOTH LISTEN 11 /usr/bin/forkserver**	This is the simplest Bluetooth RFCOMM server example. Use, for example, **btserver** as a client to test this example. This example waits for a full line from the client, echoes is back and then exits.
btlogger	**SET BLUETOOTH LISTEN 11 /usr/bin/btlogger /tmp/logfile**	This is a simple Bluetooth RFCOMM server example, which logs lines received from the connected client, and answers with "ACK". Use, for example, **btserver** as a client to test this example.

Example	Usage	Purpose
btserver	**/usr/bin/btserver - channel** for server mode (if no forkserver is running), **/usr/bin/btserver <bdaddr of btserver in server mode or forkserver> channel** for client mode	This is an advanced iWRAP client example, which can run both as an RFCOMM server, when it works as **forkserver**, or as a client, when it sends "YooHoo" to remote server, waits, displays the response, and quits).
ledtest	**/usr/bin/ledtest**	I/O: LED example.
m2n	**echo testmessage | /usr/bin/m2n**	This is a Machine-2-Network (M2N) example. For actual remote connection, **syslog** must be configured to log to a remote syslog server.
www	Browse to http://wrap-ip-address/example.html	Demonstration of the web server capabilities.
makesms	Browse to http://wrap-ip-address/send.html. Notice that this example assumes that WRAP SMS Gateway is up and running (see Section 3.4.3).	This example demonstrates WRAP SMS Gateway by sending SMS messages with it.
setup-helloworld	Install the generated WPK and navigate in WWW Setup	This example demonstrates how to add a new helloworld submenu to the WWW Setup, with two menu items that change the variables in `/etc/sysconfig/helloworld` file.
lottery	Install the generated WPK and uncomment **exec** example in `/etc/obexsender.conf`	This example demonstrates the how ObexSender responses can be generated from an application.

Table 6-1. Examples, Their Usage and Purpose

6.3.2. Creating a New Project

To start a new project, you must create a new subdirectory in your Development Kit's directory (`asdk/`) and add your application source files and `Makefile` to that directory.

A project skeleton can be automatically created by using the Bluegiga SDK AppWizard. Just give the **make appwiz APP=dir/to/newapp** command in the Development Kit's top level directory (`asdk/`). A "hello world" example ANSI C project is then created.

To use C++ compiler, replace `$(do_link)` with `$(do_link_cc)` in `Makefile`.

The details of the compile process and variables you may need to modify before compiling your application, such as `CFLAGS`, `LDFLAGS` and `CXXFLAGS`, can be seen in file `asdk/Rules.mak`.

Now you have a new project waiting for coding. To compile the project, run **make** in the `asdk/dir/to/newapp` directory.

The build system also creates the installation packets to all supported architectures (`asdk/dir/to/newapp/_{arch}/newapp-timestamp.arch.wpk`), which can be transferred to the `/tmp/obex` directory of Access Server or Access Point from where it is installed automatically.

6.3.3. Building from the Command Line

The Bluegiga Development Kit uses the ARM port of the GNU bintools and compilers to build applications. If you are not familiar with Linux development, use the method explained in the previous section instead of writing your own makefiles.

If you still want to use your own development environment, there are two minor issues to remember:

1. Tools are prefixed with **arm-linux-**, so for calling the **gcc** C-compiler, you must call **arm-linux-gcc**, and so on.
2. Tools are located in `/usr/local/arm/{sdk_version}.{architecture}/bin/` directory, which is not in PATH by default.

6.3.4. Transferring an Application to Access Server or Access Point

To run an application on Access Server or Access Point, it must first be transferred to it. There are several ways of doing this (see Section 2.3.3). The most convenient ways in conjunction with software development are discussed in the following subsections.

6.3.4.1. Transferring an Application Using SCP or SFTP

An SCP transfer is done with a single command. In the following example, `myapp` is transferred to the `/tmp` directory in Access Server or Access Point:

```
$ scp myapp root@<wrap-ip-address>:/tmp
root@<wrap-ip-address>'s password: buffy (not echoed back)
/path/to/myapp/myapp  100%    20KB    20.0KB/s    00:00
$
```

An SFTP transfer is almost similar, but the command procedure resembles an FTP session (FTP can also be used if the FTP server is enabled):

```
$ sftp root@<wrap-ip-address>
Connecting to <wrap-ip-address>...
root@<wrap-ip-address>'s password: buffy (not echoed back)
sftp> cd /tmp
sftp> put myapp
Uploading myapp to /dev/shm/tmp/myapp
/path/to/myapp/myapp  100%    20KB    20.0KB/s    00:00
sftp> quit
$
```

6.3.4.2. Using SSHFS

With SSHFS, the Access Server or Access Point filesystem can be securely mounted to be a part of the development host's filesystem.

To download and install SSHFS, visit http://fuse.sourceforge.net/sshfs.html. After installation you can mount the whole filesystem and copy the `myapp` application to the `/tmp` directory in Access Server by using the following commands:

```
$ mkdir mnt
$ sshfs root@<wrap-ip-address>: mnt
root@<wrap-ip-address>'s password: buffy (not echoed back)
$ cp myapp mnt/tmp
$ fusermount -u mnt
$
```

6.3.4.3. Transferring an Application Using Terminal Software

If your Access Server is not connected to a LAN, you can use terminal software of your choice to transfer data to Access Server. This method is not possible with an Access Point that does not have a management console.

You can install a X/Y/Zmodem protocol application to Access Server from software package `rzsz` (see Section 3.2, which allows you to transfer data over the console using almost any terminal software available:

1. Connect your computer to the Access Server management UART using a cross-over serial cable, and start your terminal software (use settings: 115 200bps, 8 data bits, no parity, 1 stop bit).
2. Change your working directory to where you want to upload your application, and run the Xmodem application with your application name as a parameter.
3. Start Xmodem send from your terminal software.

Example 6-1. Transferring Files with Xmodem

```
[root@wrap /] cd /tmp
[root@wrap /tmp] rx testapp
rx: ready to receive testapp.
now start xmodem (checksum, not CRC) send from your terminal
[root@wrap /tmp]
```

If you want to save the application to `/usr/local/bin` (on the flash file system), you will have to replace **cd /tmp** with **cd /usr/local/bin** (and possibly create the directory, if it does not exist). To examine Access Server and Access Point directory structure, please see Appendix A.

6.3.4.4. Using Other Transfer Methods

NFS and CIFS mounts (instructions in Section 3.8) can also be used.

6.3.5. Running an Application Transferred to Access Server or Access Point

To run the application you just transferred to Access Server or Access Point, you need access to the command prompt, either using terminal software connected to the Access Server management UART or using the SSH connection to Access Server or Access Point (log in as user **root** and the root password, which is **buffy** by default).

At command prompt, change the directory to where your application is located and change file permissions so that it can be executed, then run it.

Example 6-2. Running an Application

```
[root@wrap /] cd /tmp
[root@wrap /tmp] chmod 755 testapp
[root@wrap /tmp] ./testapp
```

6.3.6. Using GNU Project Debugger (GDB)

You can use GDB, the GNU Project debugger, to debug applications in Access Server or Access Point. This requires that you install `gdbserver-*.wpk` package to Access Server or Access Point. You can find it from SDK CD or from `asdk/gpl/gdbserver/_{arch}` directory on your development PC, or install it with command **wpkgd install gdbserver** from Bluegiga software repository if your device is connected to Internet.

To debug an application it has to be compiled with debugging options enabled and stripping disabled. This can be done by modifying the `CFLAGS` variable. If you are using C++ compiler, you need to modify `CXXFLAGS` variable instead. You can do this by modifying the `Makefile` for your project:

```
# Makefile

       SDKBASE=/home/user/asdk
       ifdef SDKINSTALL
       hello:
              tar xf $(CURDIR)/_$(HW)/$@-[0-9]*.$(HW).wpk -C $(SDKBASE)/tmp/install
       endif # SDKINSTALL

       include $(SDKBASE)/Rules.mak
       CFLAGS := $(subst -Os -s„$(CFLAGS)) -ggdb
       _$(HW)/hello: hello.c
...
```

After you have compiled your application with these options and transferred it to Access Server or Access Point, you can start debugging the application as follows (example from Access Point, SDK version 4.0, `lt` architecture):

1. Start **gdbserver** on Access Point. For example:

   ```
   gdbserver :6789 /tmp/hello
   ```

2. Start **arm-linux-gdb** on development PC. For example:

```
/usr/local/arm/4.0.lt/bin/arm-linux-gdb \
-ex 'set solib-absolute-prefix \
             /usr/local/arm/4.0.lt/arm-unknown-linux-gnueabi/sys-root' \
-ex 'target extended-remote Access_Point_IP:6789' \
hello
```

3. Run the application by using the **continue** command.

You can also use Data Display Debugger (DDD), a graphical front-end to GDB. Start it as:

```
ddd --debugger "/usr/local/arm/4.0.lt/bin/arm-linux-gdb \
-ex 'set solib-absolute-prefix \
/usr/local/arm/4.0.lt/arm-unknown-linux-gnueabi/sys-root' \
-ex 'target extended-remote Access_Point_IP:6789'" \
hello
```

6.3.7. Native SDK

It is also possible compile applications for Access Server or Access Point using a native toolchain. To use it, copy files `sdk-{architecture}.iso` (see Table 3-2 for description of supported architectures) and `sdkmount.noarch.wpk` from directory `sdk/native` in the Bluegiga SDK DVD-ROM (or ISO image) to the root directory of an USB memory dongle, and insert it to Access Server's or Access Point's USB port. (You can also use Compact Flash memory card with Access Server for this purpose in similar manner). The native SDK is automatically mounted and you can start using the compiler (**gcc**) in Access Server or Access Point. All tools now available can be found in directory `/usr/sdk/bin`.

> **Note:** You can generate the `sdk-{architecture}.iso` files with command **make sdk** in SDK directory (`/home/user/asdk`). You will need to have `mkisofs` installed for the command to succeed. Generated files can then be found in `/home/user/asdk/tmp` directory.

Chapter 7. iWRAP - The Bluetooth API

The Bluetooth service in Access Server and Access Point is controlled through a TCP socket interface, called iWRAP. The first iWRAP server is listening on port 10101. In the case of Access Server 2292 and 2293, the second iWRAP server is listening on port 10102, and the third one of Access Server 2293 is listening on port 10103. All commands to the iWRAP server and replies from the server are plain ASCII strings ending in CR+LF ("\r\n"). Commands and replies are not case sensitive.

When connecting to iWRAP, you must first wait for the `READY.` prompt. Do not send any commands prior to this.

By default the connection to iWRAP is protected by a password. The default password is **`buffy`**. The password can be disabled or changed. For more information, see the SET command. If the password is enabled, it must be sent first, immediately following the `READY.` prompt. Otherwise all commands will fail with an error code.

Some replies are broadcasted to all clients by the iWRAP server. If you see something that you have not requested or the reply is not intended to you, simply ignore the reply.

In the following examples, **`bold lines`** are commands sent by the client to the iWRAP server and `normal lines` are replies received from the iWRAP server by the client.

7.1. Terms

Bluetooth address (bdaddr) consists of six hex digits separated by a colon. For example, "00:07:80:80:bf:01". With commands requiring a Bluetooth address, you can also use the Bluetooth friendly name instead, but only if it has been retrieved before with the **NAME** command.

Bluetooth RFCOMM channels are numbered from 1 to 30. In Access Server, Serial Port Profile is assigned to channel number two, Object Push Profile and File Transfer Profile to channel number three, and LAN Access Profile is on channel number four. The other channels are free for user applications.

Link Identifier (link_id) is a number from 0 to 99. It is used to identify established Bluetooth connections.

7.2. Starting iWRAP

Normally, the iWRAP servers are started automatically upon power-up. You can restart them manually (for example, to apply changes made to the iWRAP settings with the **setup** application without rebooting the system). To restart the servers manually, execute the startup script with option **restart**:

```
[root@wrap /] service bluetooth restart
```

When the iWRAP servers start up, they use the settings configured with the **setup** application. You can put additional iWRAP commands to `/etc/bluetooth.conf` file. Commands in that file are processed as the last task every time the iWRAP server is started.

7.3. Writing iWRAP Applications

There are two approaches when writing an iWRAP server program (a program accepting incoming calls) for Access Server and Access Point, both having different advantages and disadvantages:

1. Forklistener
2. iWRAP Client

Note: When writing a client program (that is, a program making an outgoing call), you have to use iWRAP.

7.3.1. Forklistener

This is a standard program reading data from standard input and writing output to standard output. See forkserver for an example of this kind of program.

Advantages:

- Easy to write.
- Very robust for simple services.
- You do not have to understand Bluetooth or iWRAP.

Disadvantages:

- Your program is started and stopped for every incoming connection.
- If there are multiple connections, it is not possible to communicate to an external program through one socket.
- You cannot use stdout for debugging; you must use syslog or a log file.
- iWRAP's advanced features are not available: powermodes, MSC, SDP, inquiry, ...

To setup a forklistener, see the **SET** command.

7.3.2. iWRAP Client

iWRAP client is a program communicating with the iWRAP server through control and data sockets. See btserver for an example of this kind of program.

Advantages:

- The disadvantages with forklistener do not apply.

Disadvantages:

- More complex than forklistener.
- You must have basic knowledge about Bluetooth and iWRAP.

- If you have multiple iWRAP clients (which is usually the case), some kind of IPC is usually needed. For example, if client #1 uses CALL command, client #2 may not do the same before client #1 receives RINGING reply. The easiest way to do this is to use LOCK command.

7.4. btcli - iWRAP Command Line Interface Utility

You can send iWRAP commands from the command line by using the **btcli** application.

Usage:

`btcli [options] command`

To see the command options, enter the **btcli --help** command.

The specified command is sent to iWRAP server (the first server at port 10101 by default) and all replies are echoed to the standard output. The application waits and prints the replies for a certain amount of time (6 seconds by default) and exits.

7.5. iWRAP Commands

INFO

`INFO` — Get basic info

Synopsis

INFO

Description

INFO is used to retrieve version information on the iWRAP server, in the same format as presented by the `READY.` prompt when the iWRAP connection is opened.

Reply

```
READY. (4.0/112233445566 $ bt2.0 00:07:80:80:bf:01 0afa)
```

Note: You should handle the whole content inside parenthesis as a single information string in your applications. The format and even ordering of the fields may change in the future. Information below is provided just for debugging purposes.

Information string fields explained:

4.0/112233445566

Software version number and version control tag of this iWRAP server

bt 2.0

Bluetooth standard capability of the baseband this iWRAP server is connected to

00:07:80:80:bf:01

Bluetooth address of the baseband this iWRAP server is connected to

0afa

Firmware version (hex) of the baseband this iWRAP server is connected to

QUIT

QUIT — Close iWRAP connection

Synopsis

QUIT

Description

To close the connection to the iWRAP server, use the **QUIT** command.

Reply

There is no reply.

Example

```
READY.
QUIT
```

SET

`SET` — Change parameters

Synopsis

SET [variable [value]]

Description

The **SET** command allows you to alter various Bluetooth and iWRAP parameters. The supported variables are listed in Table 7-1. Issuing a **SET** command without parameters lists the current settings. The most commonly used parameters can be configured with the setup application or its WWW interface. The default values for those settings are defined by setup application as well. See Section B.5 for more information.

Variable	Description
BLUETOOTH BDADDR bdaddr	Our bdaddr. This is a read-only value.
BLUETOOTH NAME friendly_name	You can set your Bluetooth friendly name with this command. Others can request this name with the **NAME** command. You can use the following meta characters: **$S**: Hardware serial number, all ten digits **$s**: Hardware serial number, last three digits **$P**: iWRAP port **$p**: last digit of iWRAP port **$H**: FQDN **$h**: hostname **$b**: our Bluetooth address, last two digits **$B**: our Bluetooth address **$$**: $
BLUETOOTH READABLE mode	If enabled, some SDP result codes will have literal values instead of numeric values. 0: No (always use numeric values) 1: Yes (literal values). This is the default value.

Variable	Description
BLUETOOTH CLASS value	You can set the class-of-device value with this command.
BLUETOOTH ROLE role {policy {timeout}}	You can set the master/slave role switch preference with this command. Optionally, you can also set the link policy and link supervision timeout. The possible values for "role" are: **0**: allow when calling, do not request when answering **1**: allow when calling, request when answering **2**: do not allow when calling, request when answering The default link policy is 000f and the default link supervision timeout is 7d00. See *Bluetooth Specification* for more information on these parameters.
BLUETOOTH ENCRYPT value	This command defines whether to use Bluetooth encryption. To actually enable Bluetooth encryption, the connection must be authenticated. 0: No 1: Yes (default)
BLUETOOTH LAP value	You can set the IAC LAP value with this command. The default value is 9e8b33.

Variable	Description
BLUETOOTH PAGEMODE mode {page_timeout {page_repetition_mode {scan_activity_interval scan_activity_window {inquiry_activity_interval inquiry_activity_window}}}}	Pagemode defines whether other devices can find and call you. There are four different modes: 0: No inquiry, no paging 1: Inquiry, no paging 2: No inquiry, paging 3: Inquiry and paging The page timeout is given in hex and the default value is 2000. The default page repetition mode is 2 (R2). The default scan activity is interval 0800 and window 0012 (R1). The default inquiry activity is interval 0800 and window 0012 (R1). See the *Bluetooth Specification* for more information on these parameters.
BLUETOOTH AUTOHIDE physical logical	This command automatically hides the baseband (sets pagemode to 0) if there are more physical ACL links or logical connections than defined. Value 0 means "don't care". Default values: 7 0
BLUETOOTH AUTH * {authflags}	This command removes the default PIN code. If you are making an outgoing connection and the remote end asks for the PIN, "0000" will be sent.
BLUETOOTH AUTH * pin {authflags}	This command sets the default PIN code.
BLUETOOTH AUTH bdaddr {authflags}	This command removes the PIN code for bdaddr.

Variable	Description
BLUETOOTH AUTH bdaddr pin {authflags}	This command sets the PIN code for bdaddr. Authflags are: **--NEWPAIR** Only if we do not have linkkey yet **--REQUEST** Request this PIN from remote, do not reply with this one **--REPLY** Reply to remote requests with this PIN **--CALL** Only if making an outgoing call **--ANSWER** Only when answering to an incoming call **--RFCOMM** Call type is RFCOMM (includes FORK/PPP/...) **--BNEP** Call type is BNEP **--L2CAP** Call type is L2CAP Default authflags are all enabled, except for **--NEWPAIR**. There are three special PINs: **-** Reject without asking PIN. **--** Reject on the connection open, do not check for call types. **+** Accept without asking PIN.
BLUETOOTH PAIR bdaddr linkkey	With this command, you can manually set the linkkey for bdaddr. Note: **SET BLUETOOTH AUTH** must also be set for a value to enable encrypted connections with previously stored link keys.
BLUETOOTH PAIR bdaddr	With this command, you can manually delete the linkkey for bdaddr.
BLUETOOTH PAIREXPIRE seconds	With this command, you can set the expiration time, in seconds, for pairing information.

Variable	Description
BLUETOOTH LISTEN channel cmd {mem {delay}}	This command adds a fork-listener for the channel. When there is an incoming RFCOMM connection to the channel, the iWRAP server handles the connection by itself by forking "cmd". At least "mem" kilobytes of free memory must be available, or the connection will be rejected. After forking, the iWRAP server waits for "delay" timerticks (50ms each) before transmitting any data. The client application must modify both the stdout and stdin pipes and set NOECHO, 8BIT and all other necessary modes at the very beginning. The purpose of the "delay" parameter is to give the application enough time to do this. You can use following meta characters in "cmd": **$b**: Bluetooth address. **$c**: RFCOMM channel **$d**: 1 if CALL, 0 if RING **$P**: iWRAP port **$C**: link_id **$$**: $
BLUETOOTH LISTEN channel host:port	This command adds a forward-listener for the channel. When there is an incoming RFCOMM connection to the channel, the iWRAP server will forward it to host:port by using a raw TCP socket.
BLUETOOTH LISTEN psm L2CAP	This command adds an L2CAP listener for the psm. You need to use this command to enable incoming L2CAP connections to a custom psm.
BLUETOOTH LISTEN psm L2CAP:host:port	This command adds a forward-listener for the psm. When there is an incoming L2CAP connection to the psm, the iWRAP server will forward it to host:port by using a raw TCP socket.
BLUETOOTH LISTEN channel	This command removes a fork/forward/L2CAP listener from the channel/psm.

Variable	Description
BLUETOOTH LINK mode params	With this command, you can modify the slave's powermode according to the "mode". "params" are optional and mode-dependent. The possible values for "mode" are: 0: Active. Params: None. 1: Park: Round-robin. Params: max_beacon min_beacon sleep_after_unpark sleep_after_round Defaults: 254 160 5 30 Sleeps are specified by timerticks (50ms). 2: Park: Idle. Params: max_beacon min_beacon max_active Defaults: 512 384 6 max_active is the maximum number of active slaves. 3: Sniff: All. Params: max_interval min_interval attempt timeout Defaults: 640 426 1 8 4: Sniff: Idle. Params: idle_timeout max_interval min_interval attempt timeout Defaults: 400 640 426 1 8 idle_timeout is in timerticks (50ms). See *Bluetooth Specification* for more information on params.

Variable	Description
BLUETOOTH QOS service_type token_rate peak_bandwidth latency delay_variation	This command sets default QoS values for a new connection. The parameters are in hex. See *Bluetooth Specification* for more information on params. Defaults: 01 00000000 00000000 000061a8 ffffffff
L2CAP TIMEOUT flushto linkto	With this command, you can define the FlushTimeout and LinkTimeout for L2CAP connections. See *Bluetooth Specification* for more information on params. Defaults: 65535 40000
PPP AUTH	Do not require any PPP authentication on incoming connections.
PPP AUTH username password	Require specified username:password on incoming PPP connections.
PPP CHANNEL channel	Our PPP (LAN Access Profile) channel. The iWRAP server handles this channel internally. If you change this, remember to modify the SDP record as well. Use zero value to disable the LAN Access Profile.
PPP DEFAULTROUTE value	This setting controls whether the iWRAP server should modify the defaultroute setting. There are four different modes: 0: Do no alter defaultroute 1: Set defaultroute according to the outgoing PPP 2: Set defaultroute according to the incoming PPP 3: Set defaultroute according to all PPP calls
PPP WINHANDSHAKE seconds	Timeout to wait for the Windows RAS handshake.
PPP IP ipaddr/mask	This command sets the network IP range for PPP clients.

Variable	Description
PAN ENABLE mask	This command controls incoming PAN connections. The mask is a sum of the following values: **1**: Allow incoming PAN-PANU connections. **2**: Allow incoming PAN-GN connections. **4**: Allow incoming PAN-NAP connections. **8**: Enable zeroconf for incoming PAN-PANU connections. **16**: Enable zeroconf for outgoing PAN-PANU connections. Default: 0
CONTROL AUTOEXEC cmd	Run the **CALL** command, and rerun it when the call is disconnected. Example: **SET CONTROL AUTOEXEC CALL bdaddr PAN-NAP PAN-NAP**
CONTROL PASSWORD	Do not require a password from iWRAP clients.
CONTROL PASSWORD pass {--LOCAL}	Enable password. iWRAP clients must send this password before giving any other command. The password is case sensitive. With an optional **--LOCAL** parameter, clients connecting from localhost are accepted without a password.
CONTROL PING seconds	If this setting is enabled (seconds > 0), the iWRAP server sends a `PING` reply to all iWRAP clients. You have to reply to it with **PONG** or the connection will be closed.
CONTROL WRITETIMEOUT timeticks	With this command, you can set in timeticks (1/20s) how long iWRAP tries to write to the datasocket if it's blocked before giving up and closing the connections.
CONTROL AUTOSAVE what filename	If this setting is enabled, the system automatically saves settings to a file when they change. See the **SAVE** command for possible "what" values.
link_id MSC value	Set MSC for link_id to value. See *ETSI TS 101 369 (GSM 07.10)* for more information.
link_id ACTIVE	With this command, you can set the powermode for a link_id to active.

Variable	Description
link_id PARK params	With this command, you can set the powermode for link_id park. Required "params" are: avg_beacon or max_beacon min_beacon See *Bluetooth Specification* for more information on params.
link_id HOLD params	With this command, you can set the link's powermode to hold. Required "params" are: avg max min See *Bluetooth Specification* for more information on params.
link_id SNIFF params	With this command, you can set the powermode for a link_id to sniff. Required "params" are: avg_interval or max_interval min_interval or max_interval min_interval attempt or max_interval min_interval attempt timeout The default attempt is 1, the default timeout is 8. See *Bluetooth Specification* for more information on params.
link_id QOS service_type token_rate peak_bandwidth latency delay_variation	With this command, you can set the link's QoS values. The parameters are in hex. See *Bluetooth Specification* for more information on params.
link_id MASTER	With this command, you can switch the role to master.
link_id SLAVE	With this command, you can switch the role to slave.

Table 7-1. Supported Parameters for iWRAP SET Command

Reply

Issuing **SET** without parameters lists all current settings. When there are parameters, there is no reply.

Example

```
READY.
SET BLUETOOTH NAME Buffy
SET BLUETOOTH PAGEMODE 3
SET BLUETOOTH READABLE 1
SET BLUETOOTH CLASS 020300
SET BLUETOOTH ROLE 0
SET BLUETOOTH ENCRYPT 0
SET BLUETOOTH PAGEMODE 3
SET BLUETOOTH AUTH * 1234
SET BLUETOOTH AUTH 00:07:80:80:bf:01 4242
SET BLUETOOTH AUTH *
SET BLUETOOTH PAIREXPIRE 600
SET BLUETOOTH LISTEN 1 /bin/login 200
SET BLUETOOTH LISTEN 2 "my/own/command with parameters" 100 5
SET BLUETOOTH LISTEN 3
SET PPP DEFAULTROUTE 0
SET PPP AUTH buffy willow
SET PPP AUTH
SET PPP CHANNEL 4
SET PPP WINHANDSHAKE 10
SET PPP IP 192.168.166.0/24

SET 0 MSC 8d

SET CONTROL PING 60
PING
PONG

SET CONTROL PASSWORD

SET CONTROL PASSWORD buffy
<client reconnects>
READY.
SET
ERROR PASSWORD NEEDED.
<client reconnects>
READY.
buffy
SET
SET BLUETOOTH BDADDR 00:07:80:80:bf:01
SET BLUETOOTH NAME Buffy
SET PPP AUTH
SET CONTROL PASSWORD buffy
SET
```

SAVE

SAVE — Save iWRAP settings

Synopsis

SAVE {what} {filename}
SAVE {what} {filename$p}

Description

The **SAVE** command writes a subset of the current settings specified with *what* to a file called *filename*.

> **Note:** It is recommended to always append *$p* to the *filename*. It is replaced with the Bluetooth baseband number. In Access Server with multiple basebands forgetting it will make saved settings to be overwritten if you issue the same command to other iWRAP servers.

To automatically save the settings when they have changed, use **SET CONTROL AUTOSAVE**. To automatically load the saved settings at startup, add **LOAD** command to the `/etc/bluetooth.conf` file.

What	Settings
AUTH	SET BLUETOOTH AUTH ...
PAIR	SET BLUETOOTH PAIR ...
BTSET	SET BLUETOOTH ..., but not AUTH or PAIR
OTHERSET	All but SET BLUETOOTH
ALL	Everything

Table 7-1. SAVE parameters

You can specify several subsets separated by commas. *ALL* is equivalent to *AUTH*,*PAIR*,*BTSET*,*OTHERSET*.

Reply

There is no reply.

Example

```
READY.
SAVE PAIR /etc/bluetooth.pair.$p
SAVE AUTH,PAIR /etc/bluetooth.security.$p
```

LOAD

LOAD — Run iWRAP command script

Synopsis

LOAD {filename}
LOAD {filename$p}

Description

The **LOAD** command runs commands from a file. This command is usually used with **SAVE** or **SET CONTROL AUTOSAVE** commands.

> **Note:** Remember to append *$p* to the *filename* in the same way as with **SAVE** or **SET CONTROL AUTOSAVE** commands to ensure that you load commands from the correct file when using Access Server with multiple basebands.

Reply

There is no reply.

Example

```
READY.
LOAD /etc/bluetooth.security.$p
SET CONTROL AUTOSAVE AUTH,PAIR /etc/bluetooth.security.$p
```

PING

PING — Ask if the connection is alive

Synopsis

PING

Description

The **PING** command can be used to check that the connection to the iWRAP server is alive.

The iWRAP can also send the PING to the client application. In that case, you must reply with the **PONG** command.

Reply

PONG

Example

```
READY.
PING
PONG

PING
PONG
```

PONG

`PONG` — Connection is alive

Synopsis

PONG

Description

The **PONG** command has to be sent back if you see a `PING` reply from the server. If you do not answer, the connection will be closed after a few seconds.

Reply

There is no reply.

Example

```
READY.
PING
PONG
```

ECHO

ECHO — Send a message to other iWRAP clients

Synopsis

ECHO {data}

Description

This command broadcasts its parameters to all iWRAP connections, including the one that sent the command.

Reply

```
ECHO data
```

Example

```
READY.
ECHO Hello world!
ECHO Hello world!
```

LOCK

LOCK — Lock other iWRAP clients

Synopsis

LOCK

Description

This command locks all other iWRAP connections to the iWRAP server in question, allowing commands only from this one. This includes all the **PING**s and **PONG**s too. Be polite and do not lock it for a long time.

Reply

There is no reply.

Example

```
READY.
LOCK
UNLOCK
```

UNLOCK

UNLOCK — Unlock other iWRAP clients

Synopsis

UNLOCK

Description

This command opens the lock created by using the **LOCK** command.

Reply

There is no reply.

Example

```
READY.
LOCK
UNLOCK
```

SHUTDOWN

SHUTDOWN — Close iWRAP server

Synopsis

SHUTDOWN

Description

To close the iWRAP server, you can use the **SHUTDOWN** command. This also immediately closes all active connections.

Reply

There is no reply.

Example

```
READY.
SHUTDOWN
```

SLEEP

SLEEP — Wait a second

Synopsis

SLEEP {seconds}

Description

The **SLEEP** command waits for a specified number of seconds before processing further commands.

SLEEP is only usable in rc scripts (/etc/bluetooth.conf).

Reply

There is no reply.

Example

```
READY.
SLEEP 4
```

LOG

`LOG` — Control iWRAP logging

Synopsis

LOG {mask} [target]

Description

To control iWRAP logging, you can use the **LOG** command. The required parameter *`mask`* is a decimal number specifying which message categories are logged. The value is a sum of bitmasks, described in Table 7-1.

Bitmask	Description
0x0000	Log nothing.
0x0001	Enable logging of fatal iWRAP events. This is the default.
0x0002	Enable logging of all iWRAP commands and events.
0x0004	Enable logging of iWRAP debugging messages.
0x0100	Enable logging of iWRAP connection related events (**READY**, **CALL**, **CONNECT**, **RING**, **NO CARRIER**).

Table 7-1. Log bitmask descriptions

The optional *`target`* parameter controls where the log messages are written. Possible targets are described in Table 7-2.

Text	Description
syslog	Log using syslog. This is the default.
iWRAP	Log to iWRAP connection.
/path/to/file	Log to the file specified.

Table 7-2. Log target options

Reply

There is no reply.

Example

```
READY.
LOG 0
LOG 1 syslog
LOG 2 iWRAP
LOG 7 /tmp/everything.log
```

7.6. Finding Bluetooth Devices

INQUIRY

INQUIRY — Search for other devices

Synopsis

INQUIRY [timeout] [--LAP {lap}]

Description

The **INQUIRY** command is used to search for other Bluetooth devices. The optional timeout number parameter defines the length of inquiry, in units of 1.25 seconds. The default timeout is 4 units, which is a good average value for all cases. The minimum timeout is 2 and the maximum is 10. The optional **--LAP** option specifies the used IAC LAP; the default value is 9e8b33 (GIAC).

During the inquiry, all devices are listed as soon as they are found by using the INQUIRY_PARTIAL reply. If the iWRAP server has cached the friendly name of the device found, it is also displayed. When the inquiry ends, a summary is displayed indicating how many devices were found. The summary also repeats the device information.

Reply

```
INQUIRY_PARTIAL bdaddr_of_dev_1 class_of_dev_1 "friendly name" rssi
INQUIRY_PARTIAL bdaddr_of_dev_2 class_of_dev_2 "friendly name" rssi
...
INQUIRY_PARTIAL bdaddr_of_dev_n class_of_dev_n "friendly name" rssi
INQUIRY number_of_devices_found
INQUIRY bdaddr_of_dev_1 class_of_dev_1 "friendly name"
INQUIRY bdaddr_of_dev_2 class_of_dev_2 "friendly name"
...
INQUIRY bdaddr_of_dev_n class_of_dev_n "friendly name"
```

Example

```
READY.
INQUIRY
INQUIRY 0

INQUIRY
INQUIRY_PARTIAL 00:07:80:80:bf:01 120300 "willow" -42
INQUIRY_PARTIAL 00:07:80:80:bf:02 520204 "" -66
INQUIRY 2
INQUIRY 00:07:80:80:bf:01 120300 "willow"
INQUIRY 00:07:80:80:bf:02 520204 ""
```

NAME

NAME — Find a friendly name

Synopsis

NAME {bdaddr}

Description

You can ask for the friendly name of another Bluetooth device with the **NAME** command.

Reply

```
NAME bdaddr "friendly name"
NAME ERROR bdaddr reason_code more_info
```

Example

```
READY.
NAME 00:07:80:80:bf:02
NAME 00:07:80:80:bf:02 "buffy"
NAME 00:07:80:80:bf:01
NAME ERROR 00:07:80:80:bf:01 108 HCI_ERR_PAGE_TIMEOUT
```

7.7. Bluetooth Connections

CALL

`CALL` — Connect to other device

Synopsis

CALL {bdaddr} SDP
CALL {bdaddr} {psm} L2CAP
CALL {bdaddr} {channel} RFCOMM
CALL {bdaddr} {uuid} RFCOMM
CALL {bdaddr} {channel} PPP [username password]
CALL {bdaddr} {uuid} PPP [username password]
CALL {bdaddr} {channel} WINPPP [username password]
CALL {bdaddr} {uuid} WINPPP [username password]
CALL {bdaddr} {channel} FORK {"/full/path/to/command and parameters"}
CALL {bdaddr} {uuid} FORK {"/full/path/to/command and parameters"}
CALL {bdaddr} {channel} FORK {host:port}
CALL {bdaddr} {uuid} FORK {host:port}
CALL {bdaddr} {PAN-destUUID} [PAN-srcUUID]

Description

The **CALL** command is used to make a connection to other Bluetooth devices. It returns the link identifier (with an immediate reply), which will be used in subsequent commands and replies.

> **Note:** Always check for a correct link_id before processing replies further.

You can use the special **FORK** call type to create an RFCOMM connection and automatically launch an application, which gets the RFCOMM connection bound to its standard input and output. The client application should modify both the stdout and stdin pipes and set NOECHO, 8BIT and all other necessary modes at the very beginning. See forkserver for an example application. If you specify *`host:port`* as parameters to **FORK**, the connection is automatically forwarded to a TCP/IP socket listening at specified port in specified host.

When you create Bluetooth PAN profile connections, you need to specify the destination PAN service with parameter *`PAN-destUUID`*, which can be either PAN-NAP (for a device routing traffic between PAN clients and other networks), PAN-GN (for a device routing traffic only between its PAN clients) or PAN-PANU (for a device doing only point-to-point PAN communication). Any of these can be specified as your own role with an optional parameter *`PAN-srcUUID`*. If you do not specify it, PAN-PANU is assumed.

> **Note:** There can only be one pending **CALL** at a time. You have to wait for the `RINGING` event before issuing another CALL. The `RINGING` event comes almost immediately after the **CALL**. You get the `ERROR 008` error if you try to establish another call too quickly. In that case, wait for some tens of milliseconds and retry. Receiving the `CONNECT` or `NO CARRIER` reply may take some time, for example, when the user is keying in the PIN code.
> **Note:** PPP is "raw" PPP without any special handshaking. WINPPP is a Windows RAS handshake followed by raw PPP. If you are unsure, use WINPPP.

Note: If you have retrieved the name of the remote device using **NAME** command, you can use that name instead of *bdaddr* parameter.

Reply

```
CALL link_id
RINGING link_id
```

Example

```
READY.
CALL 00:07:80:80:bf:01 SDP
CALL 0
RINGING 0
CONNECT 0 SDP

CALL 00:07:80:80:bf:01 4 PPP
CALL 1
RINGING 1
CONNECT 1 PPP

CALL 00:07:80:80:bf:02 4 WINPPP buffy willow
CALL 2
RINGING 2
CONNECT 2 PPP

CALL 00:07:80:80:bf:01 1 RFCOMM
CALL 3
RINGING 3
CONNECT 3 RFCOMM 1042

CALL 00:07:80:80:bf:01 2 FORK /bin/login
CALL 4
RINGING 4
CONNECT 4 FORK

CALL 00:07:80:80:bf:01 PAN-NAP
CALL 5
RINGING 5
CONNECT 5 PAN-NAP

CALL 00:07:80:80:bf:02 PAN-NAP PAN-NAP
CALL 6
RINGING 6
CONNECT 6 PAN-NAP

CALL 00:07:80:80:bf:02 2 FORK 127.0.0.1:23
CALL 7
RINGING 7
CONNECT 7 FORK
```

CLASS

`CLASS` — Incoming connection request indication

Synopsis

This is not a command.

Description

The `CLASS` reply indicates that a remote device tries to open a connection. The message contains remote device's Bluetooth address and class-of-device.

Reply

```
CLASS bdaddr class_of_device
```

Example

```
READY.
CLASS 00:07:80:80:bf:01 100108
```

CONNECT

CONNECT — Connected to other device

Synopsis

This is not a command.

Description

CONNECT is not a command, but rather a reply broadcast to you when **CALL** successfully establishes the connection. Remember to check that the link_id matches your **CALL**.

On RFCOMM/L2CAP connections, there is an additional parameter called port. Port refers to the TCP socket port number, which is used to send and receive data to and from the remote device. Connect to the port just like you connected to the iWRAP server. The connection is "raw", which means that no processing of incoming or outgoing data is made.

> **Note:** In the case of L2CAP connections, the data is handled as packets. Therefore, both the incoming and outgoing data must follow the "HDR+L2CAPDATA" format, where HDR is two bytes; first the low byte, and then the high byte of the L2CAPDATA packet length. L2CAPDATA contains the actual L2CAP packet.

Reply

```
CONNECT link_id SDP
CONNECT link_id RFCOMM port
CONNECT link_id L2CAP port
CONNECT link_id PPP
CONNECT link_id FORK
CONNECT link_id PAN-PANU
CONNECT link_id PAN-GN
CONNECT link_id PAN-NAP
```

Example

```
READY.
CALL 00:07:80:80:bf:01 SDP
CALL 0
RINGING 0
CONNECT 0 SDP

CALL 00:07:80:80:bf:01 LAN PPP
CALL 1
RINGING 1
CONNECT 1 PPP

CALL 00:07:80:80:bf:01 1 RFCOMM
CALL 2
RINGING 2
CONNECT 2 RFCOMM 1042
<Client can open socket connection to port 1042>
```

CALL 00:07:80:80:bf:01 2 FORK /bin/login
CALL 3
RINGING 3
CONNECT 3 FORK

CALL 00:07:80:80:bf:01 PAN-NAP
CALL 5
RINGING 5
CONNECT 5 PAN-NAP

CALL 00:07:80:80:bf:02 PAN-NAP PAN-NAP
CALL 6
RINGING 6
CONNECT 6 PAN-NAP

NO CARRIER

`NO CARRIER` — Disconnected from other device

Synopsis

This is not a command.

Description

The `NO CARRIER` reply indicates that you or the remote device closed the active connection, or that your **CALL** failed for some reason.

See Section 7.10 for the list of error codes. Field "more_info" is optional. If present, it gives you a human readable error code or some statistics about the closed connection.

Reply

```
NO CARRIER link_id ERROR reason
NO CARRIER link_id
```

Example

```
READY.
CALL 00:07:80:80:bf:01 4 PPP
CALL 0
RINGING 0
NO CARRIER 0 ERROR 104 HCI_ERR_PAGE_TIMEOUT

CALL 00:07:80:80:bf:01 1 RFCOMM
CALL 1
RINGING 1
CONNECT 1 RFCOMM 1042
NO CARRIER 1 ERROR 000 IN=42,OUT=66,ELAPSED=69
```

RING

RING — Another device is calling you

Synopsis

This is not a command.

Description

The RING reply indicates an incoming call from a remote device. As with CONNECT, on RFCOMM/L2CAP calls there is an additional "port" parameter. Open a socket to the port, if you want to serve this call. PPP and PAN calls are handled internally, which means that you do not have to do anything on them. The iWRAP server closes the connection if nobody grabs the call within 30 seconds.

Special call type REJECTED is used for information only. It is used if somebody tried to call you but was rejected, usually because of failing authentication.

Reply

```
RING link_id bdaddr channel PPP
RING link_id bdaddr channel RFCOMM port
RING link_id bdaddr psm L2CAP port
RING link_id bdaddr PAN-PANU
RING link_id bdaddr PAN-GN
RING link_id bdaddr PAN-NAP
RING link_id bdaddr REJECTED
```

Example

```
READY.
RING 0 00:07:80:80:bf:01 4 PPP

RING 1 00:07:80:80:bf:01 1 RFCOMM 1042
<Client can open socket connection to port 1042>

RING 2 00:07:80:80:bf:01 PAN-GN
```

RINGING

RINGING — Call in progress

Synopsis

This is not a command.

Description

The RINGING reply indicates that a previously initiated outgoing **CALL** is in the state where a new outgoing **CALL** can be made.

Reply

```
RINGING link_id
```

Example

```
READY.
CALL 1 00:07:80:80:bf:01 1 RFCOMM
<Making new CALL is not allowed but generates BUSY error>
CALL 1
<Making new CALL is not allowed but generates BUSY error>
RINGING 1
<Making new CALL is allowed>
CALL 2 00:07:80:80:bf:02 2 RFCOMM
<Making new CALL is not allowed but generates BUSY error>
CALL 2
<Making new CALL is not allowed but generates BUSY error>
RINGING 2
<Making new CALL is allowed>
CONNECT 1 RFCOMM 1042
<Client can open socket connection to port 1042>
CONNECT 2 RFCOMM 1043
<Client can open socket connection to port 1043>
```

CLOSE

CLOSE — Disconnect

Synopsis

CLOSE {link_id}

Description

The **CLOSE** command closes an active connection started with a CONNECT or RING. Note that closing the RFCOMM data socket connection also closes the Bluetooth connection.

Reply

There is no direct reply. NO CARRIER is replied when the connection actually closes. In special situations (for example when the other end has gone out of the range) this may take even tens of seconds, during which **LIST** command will show the connection in CLOSING state.

Example

```
READY.
CALL 00:07:80:80:bf:01 4 PPP
CALL 1
RINGING 1
CONNECT 1 PPP
CLOSE 1
NO CARRIER 1 ERROR 000
```

LIST

`LIST` — List connections

Synopsis

LIST

Description

The **LIST** command reports active connections and some statistics.

Reply

```
LIST number_of_connections
LIST link_id status type blocksize bytes_in bytes_out elapsed_time our_msc
  remote_msc bdaddr channel direction powermode role crypt child_pid hcihandle
LIST link_id status type blocksize bytes_in bytes_out elapsed_time our_msc
  remote_msc bdaddr channel direction powermode role crypt child_pid hcihandle
...
LIST link_id status type blocksize bytes_in bytes_out elapsed_time our_msc
  remote_mscbdaddr channel direction powermode role crypt child_pid hcihandle
```

Reply Values

Status values are:

- WAITING. The iWRAP server is waiting for someone to connect to the datasocket created with the RFCOMM `CONNECT` or `RING` event.
- CONNECTED. The data connection is up and running.
- CLOSING. The datasocket has been closed, and the Bluetooth connection shutdown is in progress.

Type is SDP, RFCOMM, PPP, PAN-PANU, PAN-GN, PAN-NAP, FORK or L2CAP.

Blocksize is the maximum transfer unit of the Bluetooth link; used for statistics only.

Bytes_in and bytes_out refer to the numbers of bytes transferred.

Elapsed_time is the number of seconds the connection has been up.

Msc is the link's MSC value for both ends.

Bdaddr is the Bluetooth address of the connected device.

Channel is the service channel of the connection.

Direction is either OUTGOING or INCOMING.

Powermode is ACTIVE, SNIFF, PARK or HOLD.

Role is MASTER or SLAVE.

Crypt is PLAIN or ENCRYPTED.

Child_pid is the child process ID for types PPP and FORK. The PID is zero for others.

Hcihandle is the HCI handle for this connection.

Example

```
READY.
LIST
LIST 1
LIST 0 CONNECTED RFCOMM 666 4242 100 30 8d 8d 00:07:80:80:bf:01 4
  OUTGOING ACTIVE MASTER PLAIN 0 2a
```

RSSI

`RSSI` — Link's RSSI value

Synopsis

RSSI {link_id}

Description

The **RSSI** command reports link_id's RSSI value.

Reply

```
RSSI link_id rssi
```

Example

```
READY.
CALL 00:07:80:80:bf:01 1 RFCOMM
CALL 1
RINGING 1
CONNECT 1 RFCOMM 1234
RSSI 1
RSSI 1 -60
```

TXPOWER

TXPOWER — Link's transmit power

Synopsis

TXPOWER {link_id}

Description

The **TXPOWER** command reports link_id's transmit power.

Reply

```
TXPOWER link_id power
```

Example

```
READY.
CALL 00:07:80:80:bf:01 1 RFCOMM
CALL 1
RINGING 1
CONNECT 1 RFCOMM 1234
TXPOWER 1
TXPOWER 1 -3
```

BER

BER — Link's bit error rate

Synopsis

BER {link_id}

Description

The **BER** command reports link_id's bit error rate.

Reply

```
BER link_id ber
```

Example

```
READY.
CALL 00:07:80:80:bf:01 1 RFCOMM
CALL 1
RINGING 1
CONNECT 1 RFCOMM 1234
BER 1
BER 1 0.2600
```

CLOCK

CLOCK — Link's piconet clock

Synopsis

CLOCK {link_id}

Description

The **CLOCK** command reports link_id's piconet clock.

Reply

```
CLOCK link_id clock accuracy
```

Example

```
READY.
CALL 00:07:80:80:bf:01 1 RFCOMM
CALL 1
RINGING 1
CONNECT 1 RFCOMM 1234
CLOCK 1
CLOCK 1 0009fdd7 0001
```

STATUS

`STATUS` — Status of a connection

Synopsis

This is not a command.

Description

The `STATUS` reply is used to inform you about changes in connection status. See also the **SET** command.

Reply

```
STATUS link_id MSC value
```

Example

```
READY.
STATUS 0 MSC 8d
```

GFRAME

GFRAME — Send G-Frame

Synopsis

GFRAME {link_id} {psm} {hexdata}

Description

This command sends one G-Frame (aka connectionless L2CAP data). Use link_id * to broadcast it to all active slaves in piconet.

Reply

There is no reply when sending G-Frame. Incoming G-Frames are reported as:

```
GFRAME link_id psm hexdata
```

Example

```
READY.
GFRAME 1 33 f00f00
GFRAME * 31 deadbeaf
```

7.8. Service Discovery

This section describes the commands used for Bluetooth service discovery and local SDP record manipulation. The commands and their replies use SDP UUID and attribute values, which are listed in the Bluetooth Assigned Numbers documentation. In the commands below, the most useful UUID and attribute values can, however, be replaced with keywords listed in Table 7-5. The same keywords are used in the command replies instead of numeric values, if the parameter **SET BLUETOOTH READABLE** is set to **1**.

Keyword(s)	Hex Value
SDP	0001
RFCOMM	0003
OBEX	0008
BNEP	000F
AVDTP	0019
AVCTP	0017
L2CAP	0100
PUBLICBROWSEGROUP, BROWSE, ROOT	1002
SERIALPORT, SPP	1101
LANACCESS, LAN	1102
DIALUPNETWORKING, DUN	1103
OBEXOBJECTPUSH, OBJP, OPP	1105
OBEXFILETRANSFER, FTP	1106
PAN-PANU, PANU	1115
PAN-NAP, NAP	1116
PAN-GN, GN	1117
PNPINFORMATION, DEVICEID	1200
AUDIOSOURCE	110A
AUDIOSINK	110B
AVREMOTECONTROLTARGET, RCTARGET	110C
AVREMOTECONTROL	110E
AVREMOTECONTROLCONTROLLER, RC	110F
SERVICECLASSIDLIST, SERVICECLASS, CLASS	0001
PROTOCOLDESCRIPTORLIST, DESCLIST, DESC	0004
SERVICEAVAILABILITY	0008
SUPPORTEDFEATURES, FEATURES	0311
SERVICENAME, NAME	0100
SERVICEDESCRIPTION, DESCRIPTION	0101
SECURITYDESCRIPTION	030A
NETACCESSTYPE	030B
MAXNETACCESSRATE	030C

Table 7-5. Supported Keywords for Replacing SDP UUIDs or Attributes

SDPSEARCH

SDPSEARCH — Browse SDP Records

Synopsis

SDPSEARCH {link_id} {uuid}

Description

The **SDPSEARCH** command is used to send a Service Search Request to a connected SDP server, identified with link_id. The command only supports searching for one UUID at a time (specified with the uuid parameter, 4 hex digits, or with a keyword), but several requests can be sent during the same SDP connection. However, you must wait for the reply to the previous reply before issuing a new **SDPSEARCH** command.

Reply

```
SDPSEARCH link_id number_of_handles
SDPSEARCH link_id handle_1
SDPSEARCH link_id handle_2
...
SDPSEARCH link_id handle_n
```

Example

```
READY.
CALL 00:07:80:80:bf:01 SDP
CALL 0
RINGING 0
CONNECT 0 SDP
SDPSEARCH 0 LANACCESS
SDPSEARCH 0 1
SDPSEARCH 0 00010000
CLOSE 0
NO CARRIER 0 ERROR 000
```

SDPATTR

SDPATTR — Browse SDP Records

Synopsis

SDPATTR {link_id} {handle} {attribute}

Description

The **SDPATTR** command is used to send a Service Attribute Request to a connected SDP server, identified with the link_id. The command supports requesting for one attribute value (specified with the attribute parameter, 4 hex digits, or a keyword) in one previously retrieved service entry (specified with the handle parameter, 8 hex digits), but several requests can be sent during the same SDP connection. However, you must wait for the reply to the previous reply before issuing a new **SDPATTR** command.

The reply contains the response from the SDP server in encoded form. The code characters are described in Table 7-1.

Char	Description
I	Unsigned integer (2, 4, or 8 hexadecimal digits) follows. This is often a handle, attribute, or attribute value. Attribute values are shown as text if **BLUETOOTH READABLE** is set to **1**.
I	Signed integer byte (2 hexadecimal digits) follows.
U	UUID (4 or 8 hexadecimal digits) follows. Shown as text if **BLUETOOTH READABLE** is set to **1**.
S	String follows.
B	Boolean follows.
<	Start of sequence.
>	End of sequence.
A	Alternative follows.
R	Universal Resource Locator follows.

Table 7-1. SDP Response Formatting Characters

Reply

```
SDPATTR link_id info
```

Example

```
READY.
CALL 00:07:80:80:bf:01 SDP
CALL 0
CONNECT 0 SDP
SDPSEARCH 0 LAN
SDPSEARCH 0 1
SDPSEARCH 0 00010000
SDPATTR 0 00010000 DESCLIST
```

```
SDPATTR 0 < I 0004 < < U 0100 > < U 0003 I 04 > > >
CLOSE 0
NO CARRIER 0 ERROR 000
```

SDPQUERY

SDPQUERY — Browse SDP Records

Synopsis

SDPQUERY {link_id} {uuid} {attribute}

Description

The **SDPQUERY** command is used to send a Service Search Attribute Request to a connected SDP server, identified with the link_id. The command supports requesting for one attribute value (specified with the attribute parameter, 4 hex digits, or a keyword) in all service entries containing one UUID (specified with the uuid parameter, 4 hex digits, or a keyword), but several requests can be sent during the same SDP connection. However, you must wait for the reply to the previous reply before issuing a new **SDPQUERY** command.

Reply

```
SDPQUERY link_id info
```

Example

```
READY.
CALL 00:07:80:80:bf:01 SDP
CALL 0
RINGING 0
CONNECT 0 SDP
SDPQUERY 0 LAN DESCLIST
SDPQUERY 0 < < I 0004 < < U 0100 > < U 0003 I 04 > > > >
SDPQUERY 0 1102 0100
SDPQUERY 0 < < I 0100 S "Lan Access using PPP" > >
CLOSE 0
NO CARRIER 0 ERROR 000
```

SDP bdaddr

`SDP bdaddr` — Check devices SDP

Synopsis

SDP {bdaddr} {uuid}

Description

The **SDP bddaddr** command is the most useful command for retrieving SDP information from the remote device. The command opens the SDP connection, makes the SDP query, closes the connection and replies to the client in encrypted form. The format is described with the **SDPATTR** command.

Reply

```
SDP bdaddr 0 ERROR reason
SDP bdaddr number_of_entries
SDP bdaddr info
SDP bdaddr info
...
SDP bdaddr info
```

Example

```
READY.
SDP 00:07:80:80:bf:01 SERIALPORT
SDP 00:07:80:80:bf:01 1
SDP 00:07:80:80:bf:01 < I SERVICENAME S "Serial Port" >
  < I PROTOCOLDESCRIPTORLIST < < U 0100 > < U RFCOMM I 0b > > >
```

SDP ADD

`SDP ADD` — Add entry to local SDP

Synopsis

SDP ADD {uuid [:uuid2]} {channel} {description}

Description

This command adds a new entry to iWRAP server's SDP record.

Reply

```
SDP handle
SDP handle ERROR reason
```

Example

```
READY.
SDP ADD LANACCESS 4 "Lan access"
SDP 65536

SDP ADD SERIALPORT 10 "Serial port"
SDP 65537

SDP ADD PAN-NAP 0 "PAN Network Access Point"
SDP 65538

SDP ADD L2CAP:1201 4099 "Private L2CAP for networking"
SDP 65539
```

SDP DEL

`SDP DEL` — Delete entry for local SDP

Synopsis

SDP DEL {handle}

Description

This command deletes one entry from iWRAP server's SDP record.

Reply

There is no reply.

Example

```
READY.
SDP DEL 65537
```

SDP LIST

`SDP LIST` — List local SDP

Synopsis

SDP LIST

Description

This command lists iWRAP server's SDP record entries.

Reply

```
SDP number_of_entries
SDP handle uuid channel description
SDP handle uuid channel description
...
SDP handle uuid channel description
```

Example

```
READY.
SDP LIST
SDP 1
SDP 65536 LANACCESS 4 "Lan access"
```

7.9. Example Sessions

Outgoing RFCOMM Call:

```
READY.
LOCK
CALL 00:07:80:80:bf:01 1 RFCOMM
CALL 2
RINGING 2
UNLOCK
CONNECT 2 RFCOMM 1042
STATUS 2 MSC 8d
<Client opens socket connection to port 1042 and transfers data>
CLOSE 2
NO CARRIER 2 ERROR 000
```

Incoming RFCOMM Call:

```
READY.
RING 2 00:07:80:80:bf:01 1 RFCOMM 1042
STATUS 2 MSC 8d
<Client opens socket connection to port 1042 and transfers data>
NO CARRIER 2 ERROR 000
```

7.10. Error Codes

Some commands may reply with an error code. The human-readable name of the error is displayed, if the **SET BLUETOOTH READABLE** setting has value **1**. Error code 8 (`ERROR 008`) indicates that the iWRAP server is busy executing a number of commands; there can be several client applications using the stack. Just wait a few seconds and try again. Other error codes indicate unexpected, but often only temporary, communication problems.

You can analyze the error from the numeric code. Values bigger than or equal to 900 are iWRAP errors, described in Table 7-7.

Error Code	Textual Form	Reason
900	SERVICE_NOT_FOUND	Tried to CALL a device whose SDP records do not include the requested service.
901	ALREADY_CONNECTED	Tried to CALL a device and a service channel that is already connected.
902	OUT_OF_HANDLES	Tried to CALL, but there are too many open connections.
903	INVALID_ADDRESS_<addr>	Tried to CALL a device with a friendly name that could not be found in the name cache.
904	REJECTED	An incoming call was rejected by the iWRAP server.
905	BUSY	Tried to issue SDPATTR, but another SDP request was in progress.

Error Code	Textual Form	Reason
906	BUSY	Tried to issue SDPQUERY, but another SDP request was in progress.
907	NOT_CONNECTED	Tried to CLOSE a connection handle that is not active.
908	BUSY	Tried to issue SDPSEARCH, but another SDP request was in progress.
909	INVALID_ADDRESS	Tried to NAME a device with a friendly name that cannot be found in the name cache.
90a	BUSY	Tried to issue NAME, but another NAME was in progress.

Table 7-7. iWRAP Errors

Other error codes can be analyzed as follows. For example, `NO CARRIER ERROR 465`: The number 465 is hexadecimal, the sum of 0x400 and 0x65, where 0x400 is a mask, which means that this is an RFCOMM level error (see Table 7-8). 0x65 (decimal 101) means that the RFCOMM error was a connection timeout.

Mask	Error level
0x100	HCI
0x200	L2CAP
0x300	SDP
0x400	RFCOMM

Table 7-8. Errors Masks

The error codes for each mask are listed in the following tables.

Error Code	Textual Form	Bluetooth HCI Specification Error
0x100	HCI_SUCCESS	Success (0x00)
0x101	HCI_ERR_UNKNOWN_COMMAND	Unknown HCI Command (0x01)
0x102	HCI_ERR_NOCONNECTION	Unknown Connection Identifier (0x02)
0x103	HCI_ERR_HARDWARE_FAIL	Hardware Failure (0x03)
0x104	HCI_ERR_PAGE_TIMEOUT	Page Timeout (0x04)
0x105	HCI_ERR_AUTHENTICATION_FAILED	Authentication Failure (0x05)
0x106	HCI_ERR_KEY_MISSING	PIN or Key Missing (0x06)
0x107	HCI_ERR_MEMORY_FULL	Memory Capacity Exceeded (0x07)
0x108	HCI_ERR_CONNECTION_TIMEOUT	Connection Timeout (0x08)

Error Code	Textual Form	Bluetooth HCI Specification Error
0x109	HCI_ERR_MAX_NUM_CONNECTIONS	Connection Limit Exceeded (0x09)
0x10a	HCI_ERR_MAX_NUM_SCO_CONNECTIONS	Synchronous Connection Limit to a Device Exceeded (0x0a)
0x10b	HCI_ERR_ACL_CONN_ALREADY_EXISTS	ACL Connection Already Exists (0x0b)
0x10c	HCI_ERR_COMMAND_DISALLOWED	Command Disallowed (0x0c)
0x10d	HCI_ERR_HOST_REJECTED_0D	Connection Rejected due to Limited Resources (0x0d)
0x10e	HCI_ERR_HOST_REJECTED_0E	Connection Rejected due to Security Reasons (0x0e)
0x10f	HCI_ERR_HOST_REJECTED_0F	Connection Rejected due to Unacceptable BD_ADDR (0x0f)
0x110	HCI_ERR_HOST_TIMEOUT	Connection Accept Timeout Exceeded (0x10)
0x111	HCI_ERR_UNSUPPORTED_PARAM_VALUE	Unsupported Feature or Parameter Value (0x11)
0x112	HCI_ERR_INVALID_HCI_PARAMETER_VALUE	Invalid HCI Command Parameters (0x12)
0x113	HCI_ERR_OTHER_END_TERMINATE_13	Remote User Terminated Connection (0x13)
0x114	HCI_ERR_OTHER_END_TERMINATE_14	Remote Device Terminated Connection Due to Low Resources (0x14)
0x115	HCI_ERR_OTHER_END_TERMINATE_15	Remote Device Terminated Connection due to Power Off (0x15)
0x116	HCI_ERR_CONNECTION_TERMINATE_LOCALLY	Connection Terminated by Local Host (0x16)
0x117	HCI_ERR_REPEATED_ATTEMPTS	Repeated Attempts (0x17)
0x118	HCI_ERR_PARING_NOT_ALLOWED	Pairing not Allowed (0x18)
0x119	HCI_ERR_UNKNOWN_LMP_PDU	Unknown LMP PDU (0x19)
0x11a	HCI_ERR_UNSUPPORTED_REMOTE_FEATURE	Unsupported Remote Feature / Unsupported LMP Feature (0x1a)
0x11b	HCI_ERR_SCO_OFFSET_REJECTED	SCO Offset Rejected (0x1b)
0x11c	HCI_ERR_SCO_INTERVAL_REJECTED	SCO Interval Rejected (0x1c)

Error Code	Textual Form	Bluetooth HCI Specification Error
0x11d	HCI_ERR_SCO_AIR_MODE_REJECTED	SCO Air Mode Rejected (0x1d)
0x11e	HCI_ERR_INVALID_LMP_PARAMETERS	Invalid LMP Parameters (0x1e)
0x11f	HCI_ERR_UNSPECIFIED_ERROR	Unspecified Error (0x1f)
0x120	HCI_ERR_UNSUPPORTED_LMP_PARAMETER_VAL	Unsupported LMP Parameter Value (0x20)
0x121	HCI_ERR_ROLE_CHANGE_NOT_ALLOWED	Role Change not Allowed (0x21)
0x122	HCI_ERR_LMP_RESPONSE_TIMEOUT	LMP Response Timeout (0x22)
0x123	HCI_ERR_LMP_ERROR_TRANSACTION_COLLISION	LMP Error Transaction Collision (0x23)
0x124	HCI_ERR_LMP_PDU_NOT_ALLOWED	LMP PDU not Allowed (0x24)
0x125	HCI_ERR_ENCRYPTION_MODE_NOT_ACCEPTABLE	Encryption Mode not Acceptable (0x25)
0x126	HCI_ERR_UNIT_KEY_USED	Link Key can not be Changed (0x26)
0x127	HCI_ERR_QOS_NOT_SUPPORTED	Requested QoS not Supported (0x27)
0x128	HCI_ERR_INSTANT_PASSED	Instant Passed (0x28)
0x129	HCI_ERR_PAIRING_WITH_UNIT_KEY_NOT_SUPP	Pairing with Unit Key not Supported (0x29)
0x164	HCI_ERR_ILLEGAL_HANDLE	n/a
0x165	HCI_ERR_TIMEOUT	n/a
0x166	HCI_ERR_OUTOFSYNC	n/a
0x167	HCI_ERR_NO_DESCRIPTOR	n/a

Table 7-9. HCI Error Codes

Error Code	Textual Form
0x200	L2CAP_NO_CAUSE
0x201	L2CAP_ERR_PENDING
0x202	L2CAP_ERR_REFUS_INV_PSM
0x203	L2CAP_ERR_REFUS_SEC_BLOCK
0x204	L2CAP_ERR_REFUS_NO_RESOURCE
0x2ee	L2CAP_ERR_TIMEOUT_EXTERNAL

Table 7-10. L2CAP Error Codes

Error Code	Textual Form
0x300	SDP_ERR_RESERVED
0x301	SDP_ERR_UNSUPPORTED_SDP_VERSION
0x302	SDP_INVALID_SERVICE_RECORD_HANDLE
0x303	SDP_INVALID_REQUEST_SYNTAX
0x304	SDP_INVALID_PDU_SIZE
0x305	SDP_INVALID_CONTINUATION_STATE
0x306	SDP_INSUFFICIENT_RESOURCES
0x364	SDP_ERR_UNHANDLED_CODE
0x366	SDP_ERR_TIMEOUT
0x367	SDP_ERR_NOTFOUND
0x368	SDP_INVALID_RESPONSE_SYNTAX
0x3c8	SDP_NOT_FOUND (not really an error)

Table 7-11. SDP Error Codes

Error Code	Textual Form
0x400	RFCOMM_SUCCESS
0x401	RFCOMM_ERR_NORESOURCES
0x402	RFCOMM_ERR_ILL_PARAMETER
0x464	RFCOMM_ERR_REJECTED (Connection setup was rejected by remote side)
0x465	RFCOMM_ERR_TIMEOUT (Connection timed out)
0x466	RFCOMM_ERR_NSC (Non supported command received)
0x467	RFCOMM_ERR_ILLPARAMETER

Table 7-12. RFCOMM Error Codes

If the problems persist after restarting the communication parties, please contact Bluegiga Technologies as instructed in Section 1.2.

Chapter 8. LED, Buzzer and GPIO API

Bluegiga I/O API defines how to access Access Point's LEDs and Access Server's LEDs, buzzer, and general purpose I/O.

8.1. Write and Read

Access Point's LEDs and Access Server's LEDs, buzzer and GPIO can be accessed via `/dev/led` device. Writing an upper case character to that device enables LED, buzzer or GPIO. Writing a lower case character disables it. Reading data from the device returns current status.

In Access Point, letter "c" is LED 1 and letter "b" is LED 2 (also known as "Bluetooth LED").

In Access Server, letter "a" is the buzzer, letters "b".."e" are the LEDs, "l"..."y" are GPIO pins (pins 1 and 3-15) and "z" is GPIO enable.

Example: Turn LEDs 2 and 3 on, 1 and 4 off in Access Server.

```
[root@wrap /] echo bCDe > /dev/led
```

GPIO pin 16 is the ground pin and pin 2 is Vcc. Connected device has to supply desired voltage (from 3.3V to 5.0V) to that pin.

Example: Turn GPIO 1 on and GPIO 15 off.

```
[root@wrap /] echo Ly > /dev/led
```

8.2. Configure

There are also the following configuration commands to `/dev/led` device ("\n" denotes line feed character, ASCII 10):

":INPUT letter\n" configures the specified GPIO pin(s) as input.

":OUTPUT letter\n" configures the specified GPIO pin(s) as output.

":MAP xy\n" remaps letter x to pin y.

":POWER OFF\n" turns off "system running" indicator (slowly blinking LED 1 in Access Point, LED closest to power LED in Access Server).

":POWER ON\n" turns on "system running" indicator (slowly blinking LED 1 in Access Point, LED closest to power LED in Access Server).

":RESET\n" resets the driver to default values (all off and output, except "z" is enabled).

Example: Remap "b" (Bluetooth LED) to "a" (buzzer).

```
[root@wrap /] echo ":MAP BA" > /dev/led
```

Chapter 9. Finder Protocol

Finder protocol is used to find Access Servers or Access Points using a UDP broadcast message. Finder server is listening in port 9990 for broadcast and unicast messages. The reply is unicasted to sender.

In Access Server and Access Point a finder message can be sent with command **finder**. See **finder --help** for usage. The finder server is enabled by default.

9.1. Finder Search Message

Finder search message has four bytes:

```
0x62 0x66 0x62 0x66
```

9.2. Finder Reply Message

Finder reply message has four header bytes:

```
0x66 0x62 0x66 0x62
```

Following the header bytes there is zero or more value tuples. Each tuple has format:

Field Name	Length	Description
ID	1 byte	Tuple ID, see below
Length	1 byte	Data length, in bytes
Data	Length bytes	Value for ID

Table 9-1. Finder Tuple Format

Following tuple IDs are defined:

Tuple ID	Description of Data
0x01, ProdId	Product identification string, ASCII.
0x02, Revision	Product revision string, ASCII.
0x03, HWSerial	Hardware serial number, ASCII.
0x04, IP	IP address of "nap" interface, 4 bytes.
0x05, EthMac	Ethernet MAC address, 6 bytes.
0x06, iWRAP	iWRAP information string, ASCII.
0x07, IPString	List of all IP addresses, ASCII.
0x08, Hostname	Hostname and domain, ASCII.
0x09, Description	Free description, ASCII.
0x0a, BuildTag	Software version, ASCII.
0x0b, ObexSender	Reserved for ObexSender use, ASCII.

Table 9-2. Finder Tuple IDs

Chapter 10. Advanced Use Cases

This chapter will give you advanced use cases for Access Server and Access Point. The cases listed here are not so trivial, the simple cases are already listed mostly in Chapter 7.

10.1. Making Access Server and Access Point Secure

The most important thing is to change default passwords from Setup ⟶ Security settings page.

10.2. Saving Bluetooth Pairing Information Permanently

By default, Access Server and Access Point discard pairing information after 24 hours and do not store pairing data permanently. Therefore, rebooting removes all pairing information.

To increase the pairing data timeout and to automatically store the pairing data to the permanent storage and to automatically reload the information at reboot, append the following iWRAP commands to the end of `/etc/bluetooth.conf` file (Setup ⟶ Bluetooth settings ⟶ Edit startup script in WWW Setup):

```
# Set pairing data timeout to ~370 days (in seconds)
# Note: timeout counter is restarted at reboot
SET BLUETOOTH PAIREXPIRE 32000000

# Automatically load the pairing data
LOAD /etc/bluetooth.security$p

# Automatically save the pairing data
SET CONTROL AUTOSAVE AUTH,PAIR /etc/bluetooth.security$p
```

Note: Do not forget *$p* from the filename. It is replaced with the Bluetooth baseband number. In Access Server with multiple basebands forgetting it will make security data to be overwritten by other basebands.
Note: Pairing must be done between each Bluetooth device pairs. There is no way of making a single pairing between a device and all three basebands of the WRAP 2293 Access Server.

10.3. Digital Pen

Access Server and Access Point support most of the digital pens. The examples below are for Nokia Digital Pen SU-1B but they should apply to other pens too. Visit http://techforum.bluegiga.com/ for latest information and examples about digital pen support.

To install support for digital pens you will first need to add software component `dun`. See Section 3.2 for more information about adding software components

To setup Access Server or Access Point for digital pens you have to give following iWRAP commands. The best way to do this is to append the following line to `/etc/bluetooth.conf` file (Setup ⟶ Bluetooth settings ⟶ Edit startup script in WWW Setup):

```
# Load Digital Pen emulation commands
LOAD /etc/bluetooth.pen
```

The /etc/bluetooth.pen must then be created (in WWW Setup, you can do it at Setup ⟶ Advanced settings ⟶ Edit other configuration files). It should contain the lines following the example below:

```
# Emulate a phone
SET BLUETOOTH CLASS 500204
SET BLUETOOTH LISTEN 1 "*/usr/sbin/dun"
SDP ADD DUN 1 "Digital Pen DUN"

# Add info about pen
# Note: change Bluetooth address (00:07:cf:51:f6:8e) and
# pin code (9079) to match your pen. See pen's manual
# for correct values.
SET BLUETOOTH AUTH 00:07:cf:51:f6:8e 9079 --REPLY

# Optionally add a second pen
SET BLUETOOTH AUTH 00:07:cf:51:d5:2b 6603 --REPLY

# The following line is optional, and will cause
# Access Server to reject all other incoming Bluetooth connections
SET BLUETOOTH AUTH * - --NEWPAIR
```

After these settings you can pair and use the digital pen with Access Server or Access Point just like you would use it with a phone. Both modes, receiving pictures to Access Server, and external server via dialup, are supported.

10.4. OpenVPN

This chapter explains how to create a secure network between your Access Server or Access Point and a PC running Windows OS. This is done using Virtual Private Networking (VPN) and the particular software in use is OpenVPN, which is open source software and is available for everyone without charge. VPN creates a secure tunnel between Access Server and a PC, which enables you, for example, to control a GPRS connected Access Server in a remote location.

10.4.1. Prerequisites

First, download OpenVPN from http://openvpn.se/. A normal OpenVPN version using plain command line interface is available in http://openvpn.net/download.html. The basic instructions naturally apply for both versions, since the actual software is the same. OpenVPN GUI is only available for Windows OS.

This guide relies on material provided in http://openvpn.net/. If you want more specific information on features described here or other features OpenVPN provides, please visit http://openvpn.net/howto.html.

10.4.2. Installing OpenVPN

In Windows, execute the installation file and wait until it is complete. There should be no need for reboot. After this, the OpenVPN icon appears in the system tray. Right-click the icon and you can see the available options

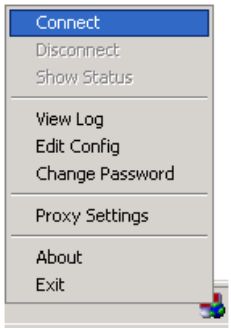

Figure 10-1. OpenVPN GUI Options Menu

For Access Server, you must install `openvpn` software component. See Section 3.2 for more information about installing software components. For Access Server and Access Point, graphical user interface is not available.

You can check that `openvpn` is installed by going to Setup ⟶ Advanced settings ⟶ System information ⟶ List installed software components. If you can see openvpn in this list, the installation is complete.

10.4.3. Creating Certificates and Keys

In this chapter, we create the necessary files to ensure privacy in the VPN, that is, we will establish a Public Key Infrastructure (PKI). The PKI consists of:

- A master Certificate Authority (CA) certificate and key which is used to sign each of the server and client certificates.
- A separate certificate (also known as a public key) and private key for the server and each client.

OpenVPN uses bi-directional authentication, which means that both server and client will authenticate each other using certificates before the connection is considered safe.

To create the files we will use a set of scripts bundled with OpenVPN for Windows. To see how the same thing is done in Linux, see http://openvpn.net/howto.html#pki.

In Windows, open up a Command Prompt window and go to `\Program Files\OpenVPN\easy-rsa`. Run the following batch file to copy configuration files into place (this will overwrite any existing `vars.bat` and `openssl.cnf` files):

```
init-config
```

Now, edit the `vars` file (called `vars.bat` on Windows) and set the *`KEY_COUNTRY`*, *`KEY_PROVINCE`*, *`KEY_CITY`*, *`KEY_ORG`*, and *`KEY_EMAIL`* parameters. Do not leave any of these parameters blank.

```
vars
clean-all
build-ca
```

The **build-ca** builds the certificate authority (CA) certificate and key by invoking the interactive **openssl** command:

```
ai:easy-rsa # ./build-ca
Generating a 1024 bit RSA private key
............++++++
...........++++++
writing new private key to 'ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FI]:
State or Province Name (full name) [NA]:
Locality Name (eg, city) [ESPOO]:
Organization Name (eg, company) [OpenVPN-TEST]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:OpenVPN-CA
Email Address [me@myhost.mydomain]:
```

Note: In the above sequence, the most queried parameters were defaulted to the values set in the `vars` or `vars.bat` files. The only parameter which must be explicitly entered is the *`Common Name`*. In the example above, we have used "OpenVPN-CA".

Next, we will generate a certificate and private key for the server:

```
build-key-server server
```

As in the previous step, most parameters can be defaulted. When the *`Common Name`* is queried, enter "server". Two other queries require positive responses, "Sign the certificate? [y/n]" and "1 out of 1 certificate requests certified, commit? [y/n]".

Generating client certificates is very similar to the previous step:

```
build-key client
```

If you want to use many clients, then you could use, for example, the following commands:

```
build-key client1
build-key client2
build-key client3
```

In this case, remember that for each client, make sure to type the appropriate *`Common Name`* when prompted, that is "client1", "client2", or "client3". Always use a unique common name for each client.

Next, we will create Diffie Hellman parameters that must be generated for the OpenVPN server:

```
build-dh
```

The output is as follows:

```
ai:easy-rsa # ./build-dh
Generating DH parameters, 1024 bit long safe prime, generator 2
This is going to take a long time
.................+.......................................
...................+.............+.................+........
.................................
```

Now you can find the generated keys and certificates in the `keys` subdirectory. The final step in the key generation process is to copy all files to the machines which need them, taking care to copy secret files (`server.key` and `client.key`) over a secure channel.

10.4.4. Creating Configuration Files

Both the server and client devices must have certain configuration files for OpenVPN to determine, for example, which IP addresses to use. In this chapter, we will create a basic configuration file for OpenVPN server and client. We'll make the PC as server and Access Server as the client. An example configuration files can be found here: http://openvpn.net/howto.html#examples. In our example, we use most of the setting described in these files.

> **Note:** The configuration files can be named, for example, `server.conf` and `client.conf` in a Linux system. On Windows they would be named `server.ovpn` and `client.ovpn`, where the file extension is different.

10.4.4.1. Server Configuration File

There are lots of configuration options that can be used with OpenVPN, but this guide only covers the basic approach to set up a working VPN with minimal effort. The minimal server configuration file is like following:

```
port 1194
proto udp
dev tun
ca "C:\\Program Files\\OpenVPN\\config\\ca.crt"
cert "C:\\Program Files\\OpenVPN\\config\\server.crt"
key "C:\\Program Files\\OpenVPN\\config\\server.key"
dh "C:\\Program Files\\OpenVPN\\config\\dh1024.pem"
server 172.30.203.0 255.255.255.0
ifconfig-pool-persist C:\\Program Files\\OpenVPN\\config\\Logs\\ipp.txt
keepalive 10 120
persist-key
persist-tun
status C:\\Program Files\\OpenVPN\\config\\Logs\\openvpn-status.log
verb 3
tls-timeout 4
```

The configuration lines are explained in detail in the following:

```
port 1194
```

- Determines the TCP or UDP port that OpenVPN should listen to. For multiple OpenVPN instances on the same machine, you'll need to use a different port for each one. Make sure your firewall allows traffic through these ports.

```
proto udp
```

- Determines whether to use TCP or UDP. We have chosen UDP in our application.

```
dev tun
```

- Determines whether to use routed IP channel (tun) or an ethernet tunnel, i.e. ethernet bridging (tap). 'tap' creates a virtual ethernet adapter, while 'tun' device is a virtual point-to-point IP link. We have chosen 'tun' because of its better efficiency and scalability.

```
ca "C:\\Program Files\\OpenVPN\\config\\ca.crt"
```

- This is known as a master Certificate Authority (CA) certificate. This will be placed in both the server and client devices, it is the same for all devices. Since the server is a Windows machine, we need to use double backslashes (\\ ) in pathnames. In Linux system one slash (/) is used.

```
cert "C:\\Program Files\\OpenVPN\\config\\server.crt"
```

- This is the certificate (a.k.a public key) for the server device.

```
key "C:\\Program Files\\OpenVPN\\config\\server.key"
```

- This is the private key for the server device and it should be kept secret.

```
dh "C:\\Program Files\\OpenVPN\\config\\dh1024.pem"
```

- This file refers to Diffie-Hellman key exchange, which is a cryptographic protocol that allows two devices that have no prior knowledge of each other to establish a shared secret key over an insecure connection.

```
server 172.30.203.0 255.255.255.0
```

- Here we create the VPN subnet. In this example, the server will take 172.30.203.1 for itself, the rest will be left for clients to use. Each client will be able to reach the server on 172.30.203.1.

```
ifconfig-pool-persist C:\\Program Files\\OpenVPN\\config\\Logs\\ipp.txt
```

- This file maintains a record of client <-> virtual IP address associations. If OpenVPN goes down or is restarted, reconnecting clients can be assigned the same virtual IP address that was previously assigned.

```
keepalive 10 120
```

- This feature causes ping-like messages to be sent back and forth over the link so that each side knows when the other side has gone down. The default parameter "10 120" makes ping occur every 10 seconds and remote peer is assumed down if no ping is received within 120 seconds.

```
persist-key
```

- Persist features try to avoid accessing certain resources on restart that may no longer be accessible.

```
persist-tun
```

- See above.

```
status C:\\Program Files\\OpenVPN\\config\\Logs\\openvpn-status.log
```

- OpenVPN outputs a short status description to this file showing current connections. This file is truncated and rewritten every minute.

```
verb 3
```

- This sets the verbosity level of the log file.
  - 0 is silent, except for fatal errors
  - 4 is reasonable for general use
  - 5 and 6 can help to debug connection problems
  - 9 is extremely verbose

```
tls-timeout 4
```

- Packet retransmit timeout on TLS control channel if no acknowledgment from remote end within n seconds (n = 4 in this example).

10.4.4.2. Client Configuration File

Just like with the server configuration file, we will describe here the basic client settings needed in our example configuration:

```
client
dev tun
proto udp
remote 192.168.42.1 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca /usr/local/openvpn/ca.crt
cert /usr/local/openvpn/conf/client1.crt
key /usr/local/openvpn/conf/client1.key
verb 3
```

The configuration lines are explained in detail in the following:

```
client
```

- Here we specify that we are a client and that we will be pulling certain config file directives from the server.

```
dev tun
```

- This setting is the same as in the server configuration file. Use the same setting you are using in the server.

```
proto udp
```

- This setting is the same as in the server configuration file. Use the same setting you are using in the server.

```
remote 192.168.42.1 1194
```

- This setting configures the hostname/IP and port of the server.

```
resolv-retry infinite
```

- Keep trying indefinitely to resolve the host name of the OpenVPN server. Very useful on machines which are not permanently connected to the Internet, such as laptops.

```
nobind
```

- Most clients do not need to bind to a specific local port number.

```
persist-key
```

- This setting is the same as in the server configuration file. Use the same setting you're using in the server.

```
persist-tun
```

- This setting is the same as in the server configuration file. Use the same setting you're using in the server.

```
ca /usr/local/openvpn/conf/ca.crt
```

- This is the same `ca.crt` file as in the server. See server config file descriptions for more information.

```
cert /usr/local/openvpn/conf/client.crt
```

- This is the certificate (a.k.a public key) for the client device.

```
key /usr/local/openvpn/conf/client.key
```

- This is the private key for the client device.

```
verb 3
```

- Sets the verbosity level of the log file.

10.4.5. Starting up VPN

First, place the configuration files in the client and server. Like in the examples, the location for these files can be, for example, `C:\Program Files\OpenVPN\config` in Windows and `/usr/local/openvpn/config` in Linux. Next, copy the authentication files (`ca.crt`, `server.crt`, `server.key`, `client.crt` and `client.key`) into the same directories.

10.4.5.1. Starting up the Server

The OpenVPN server must be accessible from the Internet:

- open UDP port 1194 on the firewall (or the TCP/UDP port you have configured), or
- set up a port forward rule to forward UDP port 1194 from the firewall/gateway to the machine running the OpenVPN server
- make sure TUN/TAP device is allowed access through firewalls

To start the OpenVPN server right-click on the `.ovpn` file on Windows and choose "Start OpenVPN on this config file" or by right-clicking the GUI icon on taskbar and start correct config file from there. It's also possible to start from command line:

```
openvpn [server_config_file]
```

Where "server_config_file" is in our Windows examples is `server.ovpn`.

A normal server startup should look like this (output will vary across platforms):

```
Sun Feb  6 20:46:38 2005 OpenVPN 2.0_rc12 i686-suse-linux [SSL] [LZO] [EPOLL] built on Feb
Sun Feb  6 20:46:38 2005 Diffie-Hellman initialized with 1024 bit key
Sun Feb  6 20:46:38 2005 TLS-Auth MTU parms [ L:1542 D:138 EF:38 EB:0 ET:0 EL:0 ]
Sun Feb  6 20:46:38 2005 TUN/TAP device tun1 opened
Sun Feb  6 20:46:38 2005 /sbin/ifconfig tun1 10.8.0.1 pointopoint 10.8.0.2 mtu 1500
Sun Feb  6 20:46:38 2005 /sbin/route add -net 10.8.0.0 netmask 255.255.255.0 gw 10.8.0.2
Sun Feb  6 20:46:38 2005 Data Channel MTU parms [ L:1542 D:1450 EF:42 EB:23 ET:0 EL:0 AF:3/
Sun Feb  6 20:46:38 2005 UDPv4 link local (bound): [undef]:1194
Sun Feb  6 20:46:38 2005 UDPv4 link remote: [undef]
Sun Feb  6 20:46:38 2005 MULTI: multi_init called, r=256 v=256
Sun Feb  6 20:46:38 2005 IFCONFIG POOL: base=10.8.0.4 size=62
Sun Feb  6 20:46:38 2005 IFCONFIG POOL LIST
Sun Feb  6 20:46:38 2005 Initialization Sequence Completed
```

10.4.5.2. Starting up the Client

We'll start the client from Linux command line:

```
openvpn [client_config_file]
```

Where "client_config_file" is in our examples `client.conf`.

A normal client startup looks similar to the server output and should end with the "Initialization Sequence Completed" message.

Now, try a ping across the VPN from the client:

```
ping 10.8.0.1
```

If the ping succeeds, you have a functioning VPN.

Chapter 11. Access Point Certification Information and WEEE Compliance

Access Point is CE approved and Bluetooth qualified v. 2.0 + EDR. It has been measured against the following specification standards: ETSI EN 300 328 v1.6.1 / EN 301 489-1/17 / EN 60950-1 / FCC parts 15.247, 15.209, 15.207, 15.109 and 15.107. Supported Bluetooth profiles are: GAP, SDAP, LAN client and server, SPP A and B, FTP client and server, ObjP client and server, PAN-PANU, PAN-GN and PAN-NAP.

Hereby, Bluegiga Technologies declares that this Access Point is in compliance with the essential requirements and other relevant provisions of Directive 1999/5/EC.

This device complies with Part 15 of the FCC Rules.

The device operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. This device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna
- Increase the distance between the equipment and receiver
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- Consult the dealer or an experienced radio or television technician for help

Warning

Changes or modifications made to this equipment not expressly approved by Bluegiga Technologies Inc. may void the FCC authorization to operate this equipment.

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. End users must follow the specific operating instructions for satisfying RF exposure compliance. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

WEEE Compliance

The crossed-out wheeled bin means that within the European Union the product must be taken to separate collection at the product end-of-life. Do not dispose of these products as unsorted municipal waste.

Chapter 12. Access Server Certification Information and WEEE Compliance

Access Server is CE approved and Bluetooth qualified v. 2.0 + EDR. It has been measured against the following specification standards: ETSI EN 300 328 v1.6.1 / EN 301 489-1/17 / EN 60950-1 / FCC parts 15.247, 15.209, 15.207, 15.109 and 15.107. Supported Bluetooth profiles are: GAP, SDAP, LAN client and server, SPP A and B, FTP client and server, ObjP client and server, PAN-PANU, PAN-GN and PAN-NAP.

Hereby, Bluegiga Technologies declares that this Access Server is in compliance with the essential requirements and other relevant provisions of Directive 1999/5/EC.

This device complies with Part 15 of the FCC Rules.

The device operation is subject to the following two conditions:

1. This device may not cause harmful interference, and
2. This device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna
- Increase the distance between the equipment and receiver
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- Consult the dealer or an experienced radio or television technician for help

Warning

Changes or modifications made to this equipment not expressly approved by Bluegiga Technologies Inc. may void the FCC authorization to operate this equipment.

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. End users must follow the specific operating instructions for satisfying RF exposure compliance. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

Any transmitter installed in the CF card slot must not exceed 4 W of e.i.r.p. To check if a particular equipment complies with this restriction, you need to know its FCC ID

number and visit the searching engine in the FCC web site in the following Internet address, where you can find the output power by the equipment in the grant of equipment: https://gullfoss2.fcc.gov/prod/oet/cf/eas/reports/GenericSearch.cfm

If this link does not work properly, please visit the FCC website (http://www.fcc.gov/) and follow the following steps to find the searching engine:

FCC website ⟶ Office of Engineering Technology ⟶ Equipment Authorization Electronic Filing ⟶ Generic Search

Please notice that the output power listed in the grant uses different units depending on the type of the equipment, e.g.:

1. The output power for 802.11a/b/g/h equipment or similar equipment approved under §15.247 or §15.407 is listed as Conducted RF power. §15.247 or §15.407 limit the e.i.r.p. to 4 W, so this restriction is fulfilled.
2. The output power for Part 22 cellular equipment is listed as e.r.p. The relationship between e.r.p. and e.i.r.p. is the following one:

   e.i.r.p. = 1.64 x e.r.p.
3. The output power for Part 24 PCS equipment is listed as e.i.r.p.
4. For other type of equipment, please consult the distributor in order to assure the restriction is fulfilled.

Note: Definitions:
Effective Radiated Power (e.r.p.) (in a given direction): The product of the power supplied to the antenna and its gain relative to half-wave dipole in a given direction.
Equivalent Isotropically Radiated Power (e.i.r.p.) (in a given direction): The product of the power supplied to the antenna and its gain relative to an isotropic antenna.

The table below is excerpted from Table 1B of 47 CFR 1.1310 titled Limits for Maximum Permissible Exposure (MPE), Limits for General Population/Uncontrolled Exposure:

Frequency Range (MHz)	Power Density (mW/cm^2)
300 - 1500	f/1500
1500 - 100000	1.0

Table 12-1. Excerpt of Table 1B of 47 CFR 1.1310

The equipment WRAP Access Server equipment transmits in the 2400 - 2483.5 MHz frequency range, so the applicable MPE limit is 1 mW/cm^2. The equipment can be provided with up to 4 Bluetooth modules WT11# (FCC ID: QOQWT11):

Under the conditions stated above MPE limits can be guaranteed as the calculation below shows:

Example 12-1. 15.247 or 15.407 Compact Flash Card with maximum allowed e.i.r.p. of 4 W

Using Equation from page 18 of OET Bulletin 65, Edition 97-01:

$S_{\text{Compact Flash card}} = \text{Prad (e.i.r.p.)}_{\text{Compact Flash card}} / 4\pi R^2 = 4000\ mW/4\pi(20\ cm)^2$

$S_{\text{Compact Flash card}} = 0.795774\ mW/cm^2$

$S_{Total} = S_{Bluetooth} + S_{Compact\ Flash\ card} = 0.003481\ mW/cm^2 + 0.795774\ mW/cm^2$

$S_{Total} = 0.799255\ mW/cm^2 < 1\ mW/cm^2$

Example 12-2. Part 22 Compact Flash Card with maximum e.r.p. of 1.5 W (Category excluded of MPE evaluation according to §2.1091)

Using Equation from page 18 of OET Bulletin 65, Edition 97-01 and considering that e.i.r.p. = 1.64 x e.r.p.:

$S_{Compact\ Flash\ card} = Prad\ (e.i.r.p.)_{Compact\ Flash\ card} / 4\pi R^2 = 1500 \times 1.64\ mW/4\pi(20\ cm)^2$

$S_{Compact\ Flash\ card} = 0.489401\ mW/cm^2$

$S_{Total} = S_{Bluetooth} + S_{Compact\ Flash\ card} = 0.003481\ mW/cm^2 + 0.489401\ mW/cm^2$

$S_{Total} = 0.492882\ mW/cm^2 < 1\ mW/cm^2$

Example 12-3. Part 24 Compact Flash Card with maximum e.r.p. of 3 W (Category excluded of MPE evaluation according to §2.1091)

Using Equation from page 18 of OET Bulletin 65, Edition 97-01 and considering that e.i.r.p. = 1.64 x e.r.p.:

$S_{Compact\ Flash\ card} = Prad\ (e.i.r.p.)_{Compact\ Flash\ card} / 4\pi R^2 = 3000 \times 1.64\ mW\ /\ 4\pi(20cm)^2$

$S_{Compact\ Flash\ card} = 0.978803\ mW/cm^2$

$S_{Total} = S_{Bluetooth} + S_{Compact\ Flash\ card} = 0.003481\ mW/cm^2 + 0.978803\ mW/cm^2$

$S_{Total} = 0.982284\ mW/cm^2 < 1\ mW/cm^2$

WEEE Compliance

The crossed-out wheeled bin means that within the European Union the product must be taken to separate collection at the product end-of-life. Do not dispose of these products as unsorted municipal waste.

Appendix A. Directory Structure

```
Directory Tree                          Type Note
==============                          ==== ====
/                                       f    whole filesystem is root writable
|-- bin                                 f
|-- boot                                f
|-- dev                                 r
|   `-- shm                             r    ramdisk
|       |-- etc                         r    resolv.conf
|       |-- tmp                         r    /tmp
|       |   |-- obex                    r    obexserver dir
|       `-- var                         r    ramdisk part of /var
|           |-- lock                    r
|           |   `-- subsys              r
|           |-- log                     r
|           |-- run                     r
|           `-- empty                   r
|-- etc                                 f    system config and init scripts
|   |-- init.d -> rc.d/init.d           l
|   |-- ppp                             f
|   |   `-- peers                       f
|   |-- rc.d                            f
|   |   |-- init.d                      f
|   |   `-- rc3.d                       f
|   |-- rc3.d -> rc.d/rc3.d             l
|   |-- ssh                             f
|   `-- sysconfig                       f
|-- lib                                 f    system libraries
|   |-- firmware                        f
|   |-- modules                         f
|   |       `-- [module directories]    f
|   |-- pppd                            f
|   `-- xtables                         f
|-- mnt                                 f    mount points
|-- proc                                p    proc filesystem
|-- root                                f    home directory of root
|-- sbin                                f
|-- sys                                 p    sys filesystem
|-- tmp -> dev/shm/tmp                  l    temporary data (ramdisk)
|-- usr                                 f
|   |-- bin                             f
|   |-- lib                             f
|   |   `-- gconv                       f
|   |-- libexec                         f
|   |-- local                           f    mount point for second flash
|   |-- sbin                            f
|   `-- share                           f
|       |-- tabset                      f
|       `-- terminfo                    f
|           |-- a                       f
|           |-- l                       f
```

```
|           |-- s                       f
|           |-- v                       f
|           `-- x                       f
`-- var                                 f
    |-- empty -> ../dev/shm/var/empty   f
    |-- lib                             f
    |   |-- btclass                     f
    |   |-- dpkg                        f
    |   |   `-- info                    f
    |   |-- obexsender                  f
    |   `-- setup                       f
    |-- lock -> ../dev/shm/var/lock     l
    |-- log -> ../dev/shm/var/log       l    log files
    |-- run -> ../dev/shm/var/run       l
    |-- spool                           f
    |   `-- cron                        f
    |       `-- crontabs                f
    |-- tmp -> ../dev/shm/var/tmp       l
    `-- www                             f
        |-- cgi-bin                     f
        `-- html                        f    WWW pages

Types
=====

f = FLASH filesystem, read/write, files will be saved on power-down
r = RAM filesystem, read/write, files will be lost on power-down
l = symbolic link
p = proc/sys filesystem, can be used to configure Linux
```

Appendix B. Setup Options

B.1. Security settings

This submenu contains the most important security settings, like passwords.

1. Root password [buffy]

   The default value for this option is "buffy".

   This is the password of the "root" user, shown in encrypted form.

   To change the password, clear the field, enter a new password and click Save.
   Saving an empty field keeps the old password.

   Please note that the new password is shown in plain text only right after
   you have saved it. Later it is only shown encrypted, and there is no way
   to decrypt it. You must either remember it or change it again to
   something you do remember.

2. Setup password [buffy]

   The default value for this option is "buffy".

   This is the password of the "root" user for WWW setup.

   To change the password, clear the field, enter a new password and click Save.
   Saving an empty field keeps the old password.

3. iWRAP password [buffy]

   The default value for this option is "buffy".

   This password is required to be entered before entering any commands when
   communicating with iWRAP.

   To change the password, clear the field, enter a new password and click Save.
   Saving an empty field keeps the old password.

   Please note that the new password is shown in plain text only right after
   you have saved it. Later it is only shown encrypted, and there is no way
   to decrypt it. You must either remember it or change it again to
   something you do remember.

   Use "-" to disable iWRAP password.

4. Allow local clients without password [Yes]

   The default value for this option is "Yes".

   If this setting is "Yes", iWRAP password is requested only from remote
   clients, not from local clients (127.0.0.1).

5. Bluetooth PIN code []

   The default value for this option is empty.

This PIN code is used when establishing connections. Up to 16 characters can be used.

If there is no default PIN code set, Access Server does not require a PIN code when establishing connections.

However, if there is no default PIN code set, but the other device requests a PIN code, "0000" is replied.

6. wpkgd autoinstall password []

The default value for this option is empty.

This is an optional password to authenticate wpk autoinstall packets (wpk packets sent to the autoinstall directory). The password is shown encrypted here, if set.

To change the password, clear the field, enter a new password and click Save.

Please note that the new password is shown in plain text only right after you have saved it. Later it is only shown encrypted, and there is no way to decrypt it. You must either remember it or change it again to something you do remember.

Use "-" do disable the password.

The password must match the authentication parameter in the "wpkg.pif" file in the wpk packet. Otherwise the packet is not processed.

Syntax in the "wpkg.pif" file:
%wpkg-auth: auth

7. wpkgd hotplug password []

The default value for this option is empty.

This is an optional password to authenticate wpk installation packets automatically run from USB memory dongles or Compact Flash memory cards. The password is shown encrypted here, if set.

To change the password, clear the field, enter a new password and click Save.

Please note that the new password is shown in plain text only right after you have saved it. Later it is only shown encrypted, and there is no way to decrypt it. You must either remember it or change it again to something you do remember.

Use "-" to disable the password.

The password must match the authentication parameter in the "wpkg.pif" file in the wpk packet. Otherwise the packet is not processed.

Syntax in the "wpkg.pif" file:
%wpkg-auth: auth

B.2. Generic settings

```
This submenu contains generic settings.
```

1. Root password [buffy]

   ```
   The default value for this option is "buffy".

   This is the password of the "root" user, shown in encrypted form.

   To change the password, clear the field, enter a new password and click Save.
   Saving an empty field keeps the old password.

   Please note that the new password is shown in plain text only right after
   you have saved it. Later it is only shown encrypted, and there is no way
   to decrypt it. You must either remember it or change it again to
   something you do remember.
   ```

2. Description of this unit [@A #@S]

   ```
   The default value for this option is "@A #@S".

   The description helps to recognize this unit in BSM, finder or in your
   own applications. The following meta tags are available:

   @A : Product identification string ("Access Server" or "Access Point")
   @a : Product revision string
   @B : Software version
   @H : Fully Qualified Domain Name (FQDN)
   @h : hostname
   @S : Hardware serial number, all ten digits
   @s : Hardware serial number, last three digits
   @@ : @
   ```

3. Use local syslog service [Yes]

   ```
   The default value for this option is "Yes".

   This option determines whether syslog logs locally to /var/log/messages or
   not.

   Set this to No if you want to log to a remote syslog server.
   ```

4. Syslog file name [/var/log/messages]

   ```
   The default value for this option is "/var/log/messages".

   Syslog file name.
   ```

5. Size of syslog file [63]

   ```
   The default value for this option is "63".

   Size of one syslog file.
   ```

6. Number of rotated syslog files [3]

   ```
   The default value for this option is "3".

   Number of rotated syslog files to keep.
   ```

7. IP address of the remote syslog server [192.168.42.1]

   ```
   The default value for this option is "192.168.42.1".

   The IP address of the device in the network to which syslog should log to.

   The remote device must be configured to accept syslogd connections. See the
   syslog documentation on the remote device for more information on how to
   configure that.
   ```

8. Swap to NFS server [No]

   ```
   The default value for this option is "No".

   Swap to NFS server.
   ```

9. Hostname and directory for NFS swap [swap.localdomain:/var/swap]

   ```
   The default value for this option is "swap.localdomain:/var/swap".

   Hostname and directory for NFS swap.
   ```

10. NFS swap size in megabytes [64]

    ```
    The default value for this option is "64".

    NFS swap size in megabytes.
    ```

11. System clock tick [10000]

    ```
    The default value for this option is "10000".

    Set the number of microseconds that should be added to the system time for
    each kernel tick interrupt (100Hz). Increasing the value by 1 speeds up
    the system clock by about 100 ppm, or 8.64 sec/day.
    ```

12. System clock frequency [0]

    ```
    The default value for this option is "0".

    Set the system clock frequency offset. Frequency gives a much finer
    adjustment than the tick option. The value is scaled such that frequency
    65536 speeds up the system clock by about 1 ppm, or 0.0864 sec/day.
    ```

B.3. Network settings

```
This submenu contains network settings.
```

1. Hostname of the unit [wrap]

   ```
   The default value for this option is "wrap".

   The hostname of Access Server. Local applications will see this
   name. This name may be changed by dynamic network configuration.
   ```

2. Domain of the unit [localdomain]

   ```
   The default value for this option is "localdomain".

   The domain name of Access Server. Local applications will see this
   name. This name may be changed by dynamic network configuration.
   ```

3. Enable ethernet cable interface [Yes]

```
The default value for this option is "Yes".

Set this option to Yes if you want to have the ethernet cable interface
enabled.

If you do not use this interface, you may disable it to slightly increase
security and system boot speed.
```

4. Time server (NTP) []

```
The default value for this option is empty.

Hostname or IP address of the time server to be connected to retrieve correct
time using the Network Time Protocol.

Leave empty to use a random selection of 8 public stratum 2 servers.
```

5. Time server (rdate) []

```
The default value for this option is empty.

Hostname or IP address of the time server to be connected at system boot to
retrieve correct time using the Time Protocol (RFC 868).

NTP client is running by default, so rdate should not be used at all.
```

6. Update current time now by NTP [/sbin/service ntpd sync]

```
Update current time now from configured NTP servers.
```

7. Zeroconf interface [nap]

```
The default value for this option is "nap".

Defines the interface in which Zeroconf is running.
Possible interface names are "nap", "gn" and "none".
```

8. DHCP client extra parameters []

```
The default value for this option is empty.

DHCP client extra parameters. Leave empty unless you are absolutely sure
what you are doing.
```

9. Enable modem interface [No]

```
The default value for this option is "No".

Set this option to Yes if you want to have the modem interface enabled.
To use the interface, a supported Compact Flash card modem or serial
modem must be attached to Access Server. Several USB modems are also
supported.
```

10. Enable Wi-Fi interface [No]

```
The default value for this option is "No".

Set this option to Yes if you want to have the Wi-Fi interface enabled
(you can use the Wi-Fi interface with a supported Compact Flash Wi-Fi
```

```
card or USB Wi-Fi dongle).

If you do not use this interface, you may disable it to slightly increase
security and system boot speed.
```

B.3.1. Default interface settings

```
Default interface settings. By default, ethernet and Bluetooth
PAN-NAP interfaces are assigned to this interface.
```

1. Use dynamic network configuration [Yes]

   ```
   The default value for this option is "Yes".

   This option determines whether or not automatic configuration of the default
   network interface (nap) using DHCP should be attempted at boot. If set to
   no, you have to manually enter IP address and other network settings.
   ```

2. IP address [192.168.42.3]

   ```
   The default value for this option is "192.168.42.3".

   The IP address of Access Server.
   ```

3. Subnet mask [255.255.255.0]

   ```
   The default value for this option is "255.255.255.0".

   The network mask of Access Server.
   ```

4. IP address of the default gateway [192.168.42.254]

   ```
   The default value for this option is "192.168.42.254".

   The IP address of the default gateway in the LAN to which Access Server
   is connected. Use "0.0.0.0" for none.
   ```

5. List of name server IPs [192.168.42.1 192.168.42.2]

   ```
   The default value for this option is "192.168.42.1 192.168.42.2".

   The IP address(es) of the name servers, separated by space.
   ```

B.3.1.1. DHCP server settings

```
DHCP server settings. Server is disabled by default.
```

1. First IP for lease block [192.168.42.4]

   ```
   The default value for this option is "192.168.42.4".

   First IP address of the lease block.
   ```

2. Last IP for lease block [192.168.42.253]

   ```
   The default value for this option is "192.168.42.253".

   Last IP address of the lease block.
   ```

3. Subnet of lease block [255.255.255.0]

   ```
   The default value for this option is "255.255.255.0".
   ```

```
Subnet mask of the lease block.
```

4. Lease time [86400]

```
The default value for this option is "86400".

Lease time in seconds.
```

B.3.2. Ethernet cable settings

```
Ethernet cable settings.
```

1. Assign to default interface [Yes]

```
The default value for this option is "Yes".

Assigns ethernet (eth0) to default interface (nap) with settings
specified in Default interface settings.

Do NOT set this to No if you do not know what you are doing. There
is a high risk that you end up with invalid network settings if you
do so.

If you need to set a static IP address to Access Server, do it
in the Default interface settings.
```

2. Use dynamic network configuration [Yes]

```
The default value for this option is "Yes".

Use dynamic network configuration (DHCP) on ethernet interface when
it is not assigned to the default interface.
```

3. IP address [192.168.43.3]

```
The default value for this option is "192.168.43.3".

IP address of the ethernet interface when it is not assigned to the
default interface and dynamic network configuration is not in use.
```

4. Subnet mask [255.255.255.0]

```
The default value for this option is "255.255.255.0".

Network mask of the ethernet interface when it is not assigned to the
default interface and dynamic network configuration is not in use.
```

B.3.3. Modem settings

```
Modem settings.
```

1. Device [/dev/ttyUSB0]

```
The default value for this option is "/dev/ttyUSB0".

Modem device for Point-to-Point Protocol Daemon (pppd) establishing
the Internet connection.

/dev/ttyUSB0 for USB modem
```

```
/dev/ttyUSB4 for USB modem (Sierra Wireless Compass(tm) 885)
/dev/ttyACM0 for USB modem (Falcom Samba 75)
/dev/ttyAT1 for user uart (Access Server only)
/dev/ttyS0 for Compact Flash modem (Access Server only)
```

2. Force connection open [No]

```
The default value for this option is "No".

If enabled, Access Server tries to ensure that Internet connection
is open by checking it every 10 minutes. If disabled, Internet
connection is opened at startup and it is not monitored later.
```

3. IP address used in force check [194.100.31.45]

```
The default value for this option is "194.100.31.45".

The IP address that Access Server pings if "Force connection open" is
enabled.
```

4. Username [blue]

```
The default value for this option is "blue".

Username for network. Contact your GSM operator for correct value.

Some examples:

Elisa/Finland:   blue
Sonera/Finland:  blue
Wataniya/Kuwait: blue
Etisalat/UAE:    Mnet
```

5. Password [giga]

```
The default value for this option is "giga".

Password for network. Contact your GSM operator for correct value.

Some examples:

Elisa/Finland:   giga
Sonera/Finland:  giga
Wataniya/Kuwait: giga
Etisalat/UAE:    Mnet
```

6. Extra parameters for pppd []

```
The default value for this option is empty.

Optional extra parameters for pppd. Use only if you know what you are doing,
for example if you want to debug PPP connections or want to use dial on
demand.
```

7. Edit pppd options file [/etc/ppp/peers/gprs]

```
Edit pppd options file.
```

8. Edit pppd connect script file [/etc/ppp/peers/gprs.connect]

```
Edit pppd connect script file. There are commented documentation about
setting SIM PIN code and changing AP name in the file.
```

9. Re-establish modem connection [/sbin/service network restart]

```
Re-establishes modem connection.
```

10. Collect modem diagnostics [/usr/sbin/supportinfo --modem]

```
This page contains all modem diagnostics information.

Include this information when sending a support request to
support@bluegiga.com
```

B.3.4. Wi-Fi settings

```
Wi-Fi settings.
```

1. Act as a Wi-Fi Access Point [No]

```
The default value for this option is "No".

This option defines whether Access Server acts as a Wi-Fi Access
Point when Wi-Fi is enabled.
```

2. ESSID []

```
The default value for this option is empty.

Access point network name (Service Set ID).
```

3. Nickname []

```
The default value for this option is empty.

The nickname, or station name.
```

4. WEP encryption key []

```
The default value for this option is empty.

WEP encryption key for Wi-Fi.

Examples:
10 hex digits:       "abcdef1234"
26 hex digits:       "1234567890abcdef1234567890"
or                   "1234-5678-90ab-cdef-1234-5678-90"
5 ASCII characters:  "s:abcde"
13 ASCII characters: "s:abcdefghijklm"

WPA support is available by installing a separate software package,
wpa-supplicant.
```

5. Extra commands for Access Point mode [/etc/sysconfig/ifup-wlan0]

```
Extra commands for Access Point mode. Use only if you know what you are
doing. Here you can, for example, add allow/reject MAC addresses with the
"iwpriv" command.
```

6. Assign to default interface [No]

```
The default value for this option is "No".
```

```
Assigns Wi-Fi to default interface with settings specified in Default
interface settings. Do not enable this unless you are sure your Wi-Fi
hardware supports bridging in the managed mode!
```

7. Use dynamic network configuration [Yes]

   ```
   The default value for this option is "Yes".

   Use dynamic network configuration (DHCP) for the Wi-Fi interface.
   ```

8. IP address [192.168.44.3]

   ```
   The default value for this option is "192.168.44.3".

   IP address of the Wi-Fi interface.
   ```

9. Subnet mask [255.255.255.0]

   ```
   The default value for this option is "255.255.255.0".

   Subnet mask of the Wi-Fi interface.
   ```

10. Collect Wi-Fi diagnostics [/usr/sbin/supportinfo --wifi]

    ```
    This page contains all Wi-Fi diagnostics information.

    Include this information when sending a support request to
    support@bluegiga.com
    ```

11. Use WPA encryption [No]

    ```
    The default value for this option is "No".

    Use WPA encryption.
    ```

12. Edit WPA configuration file [/etc/wpa_supplicant.conf]

    ```
    Edit WPA configuration file.
    ```

B.4. Applications

```
This submenu contains settings of various applications.
```

1. Default startup applications []

   ```
   Define the applications that are to be started at startup and those
   that are not.
   ```

B.4.1. Connector settings

```
Connection forwarding enables you to configure Access Server to
automatically open and maintain connections to specific Bluetooth
devices. Connection forwarding also forwards the data from the
Bluetooth connections to a defined application or IP address using
a TCP socket.
```

1. Delay between calls [20]

   ```
   The default value for this option is "20".

   This is the delay between calls to Bluetooth devices. If the call fails
   ```

```
this is the time Connector sleeps before trying to connect to the same
or the next device again.
```

2. Logfile name [-]

```
The default value for this option is "-".

Defines the path and name of the Connector log file
(for example "/usr/local/connector/connector.log").

Type "-" to use syslog.
```

3. Register to watchdog daemon [Yes]

```
The default value for this option is "Yes".

If this is enabled, Connector will reboot Access Server
automatically if Bluetooth basebands have stopped responding.
```

4. Verbosity level [0]

```
The default value for this option is "0".

Determines the verbosity level of Connector logging. The value can
be from 0 to 4. If this setting is set to "0", there will be minimal
logging and with setting "4" there will be maximum amount of logging.

WARNING! Full verbose logging (4) should be used only for debugging
purposes, since it creates a lot of logs and the flash memory can
be filled rather quickly.
```

5. Edit configuration file [/etc/connector.conf]

```
This link opens the Connector configuration file
(/etc/connector.conf) and allows you to edit it manually.

It also allows you to change the settings that are not
configurable with the Setup application.
```

6. #1 Bdaddr [-]

```
The default value for this option is "-".

Bluetooth address of remote device, for example 00:07:80:80:bf:01.
```

7. #1 Channel [-]

```
The default value for this option is "-".

Bluetooth channel or UUID where to connect.
```

8. #1 Command [-]

```
The default value for this option is "-".

This is the application or TCP/IP address and port where the connection is forwarded.

Example:

192.168.42.1:5001
```

```
/usr/local/bin/myapp
```

B.4.2. ObexSender settings

```
This submenu contains settings for ObexSender application.
```

1. Bluetooth friendly name [W$S_$p]

```
The default value for this option is "W$S_$p".

The name shown when this device is found when inquired about by other
Bluetooth devices. The following meta tags are available:

$S : Hardware serial number, all ten digits
$s : Hardware serial number, last three digits
$P : Server port
$p : Server port, last digit
$H : Fully Qualified Domain Name (FQDN)
$h : hostname
$$ : $

For example, "Server_$p" would set the Bluetooth friendly name as
"Server_1" for first baseband, "Server_2" for second baseband and
"Server_3" for third baseband.
```

2. Minimum RSSI value before sending [-80]

```
The default value for this option is "-80".

The working range of ObexSender can be configured or limited with
this setting. When ObexSender searches for devices, the RSSI
(Receiver Signal Strength Indicator) value is also measured.
This value ranges from -128 to -1.

128 to -90 means the signal strength is very weak. A connection attempt
would very likely fail.

80 to -65 means the signal strength is ok. Connection can be created.
With Class 2 devices, like most mobile phones, this means the
phone is 10-20 meters away. A Class 1 device can be even more
than 100 meters away. Please note that -65 is recommended value for
Access Server with serial number 0607239999 and smaller.

45 to -30 means the signal is very strong. The devices are most likely
very close to each other (less than a meter away). For example testing
purposes value -45 is ideal because you send only to devices very near
to Access Server. With the serial numbers of 0607239999 and smaller,
35 or -40 can also be suitable.
```

3. Whitelist RSSI limit [0]

```
The default value for this option is "0".

Determines an RSSI limit that allows remote device to be removed from
all block lists. This can be used in a way that you bring your device
very close to Access Server and it will be able to receive content
```

```
again, without waiting for OK- or FAIL-delays to pass.

For example, you have received a file successfully from ObexSender and
you then have to wait for OK-delay to pass. You have configured
whitelist RSSI limit to -45 (or -35 if serial number is less than
0607239999) and then you bring your device practically attached to
Access Server. Now you have to wait for an inquiry to pass (blue led
starts blinking and then stops). Then after a short while you should
receive content again.
```

4. Require pairing [No]

```
The default value for this option is "No".

Use pairing to inform "I want to receive files".

Enabling this means that the user must first manually pair his or her
device with Access Server.
```

5. Delete non-matching requests [Yes]

```
The default value for this option is "Yes".

This setting applies if you are using REPLY-feature of ObexSender
and you send a file to Access Server to receive specific content.
Now, if the file you sent does not match to ObexSender configuration,
the file is saved. Matching files are always deleted. Disable this
if you have some other program doing ObjP/FTP.
```

6. Register to watchdog daemon [Yes]

```
The default value for this option is "Yes".

If this is enabled, ObexSender will reboot Access Server
automatically if Bluetooth basebands have stopped responding.
```

7. Edit configuration file [/etc/obexsender.conf]

```
This link opens ObexSender configuration file
(/etc/obexsender.conf) and allows you to edit it manually.

It also allows you to change the settings that are not
configurable with the Setup application.
```

8. Inquiry and calculate hash [/usr/sbin/obexsender-hash --download]

```
Inquiry and calculate hash. This feature allows you to expand the database
of recognizable Bluetooth devices. You can use this to calculate a hash
file from a new device and send it to Bluegiga who will update and release
a new database.
```

9. Restart ObexSender [/sbin/service obexsender hardrestart]

```
ObexSender service must be restarted to activate new settings.
```

10. Base directory for content files [/usr/local/obexsender/files/]

```
The default value for this option is "/usr/local/obexsender/files/".

Base directory for content files.
```

11. Upload new content file [/usr/local/obexsender/files/]

    ```
    This link allows you to upload files into the ObexSender file directory.

    NOTE: A send rule must also be created before the files get sent.
    ```

12. Browse content files [/usr/local/obexsender/files/]

    ```
    This link allows you to browse files on the ObexSender file system.
    ```

B.4.2.1. Timeouts and delays

```
This submenu contains ObexSender timeouts and delays.
```

1. Delay between inquiries [10]

   ```
   The default value for this option is "10".

   Delay between inquiries (Bluetooth device discoveries) in seconds.
   ```

2. If previous was ok, timeout before sending again [36000]

   ```
   The default value for this option is "36000".

   If a file has been successfully sent to a device, this timeout
   (in seconds) defines when content can be sent again to the same device.
   ```

3. If previous was fail, timeout before sending again [86400]

   ```
   The default value for this option is "86400".

   If a file transmission to a device has failed or the user has declined
   the file, this timeout (in seconds) defines when ObexSender can
   send content to the same device again.
   ```

4. Delay between retrying call [120]

   ```
   The default value for this option is "120".

   When the user does not accept or reject the file, ObexSender will try
   to send the file again. This setting determines the timeout (in seconds)
   before resend occurs.

   If you wish to disable this feature you can use the same value as in
   OK-delay or FAIL-delay, that is, the two previous settings.
   ```

5. Delay after scanning [5]

   ```
   The default value for this option is "5".

   When a remote request from the user has been received, this setting
   determines how long (in seconds) ObexSender will wait until the
   response file is sent back to the user.

   The default value is 5 seconds, because some mobile phones are not
   able to receive files over Bluetooth until at least 5 seconds
   has passed from sending.
   ```

6. Tester delay [60]

   ```
   The default value for this option is "60".
   ```

```
Determines how often content is pushed to a tester device. Tester
device is a device that is always offered content, so it is not
blocked in any case.
```

7. Pair expire timeout [0]

```
The default value for this option is "0".

How long to keep the pairing. Zero means forever.
```

B.4.2.2. Log file

```
This submenu contains ObexSender log file options.
```

1. Log file name [-]

```
The default value for this option is "-".

Defines the path and name of the ObexSender log file
(for example "/usr/local/obexsender/obexsender.log").
Log file contains information about successful and unsuccessful
transmissions, timestamps and information about sent files.

Type "-" to use syslog.

You can also use "@" followed by an IP address of a log server, which
must be another Access Server running ObexSender. Example: "@192.168.43.1"

WARNINGS:

If you log to a local file you can fill up the file system
up to the point when factory reset (erasing all data) is required
to recover normal operation.

If you log to a file on a CF/USB memory card/dongle (for example
"/mnt/usb/obexsender.log"), you must take care of creating the
mount point and mounting of the device.

If you are using syslog logging, beware that the log entries are deleted
at boot or may get overwritten if the maximum number and/or size of
the syslog files is too small.
```

2. Log prefix [-]

```
The default value for this option is "-".

This prefix is put in front of every event in the log file.
Type "-" for none.
```

3. If sending was failure, log it too [Yes]

```
The default value for this option is "Yes".

If this is enabled, failed transmissions will be logged too.
```

4. Verbosity level [0]

```
The default value for this option is "0".
```

```
Determines the verbosity level of ObexSender logging. The value can
be from 0 to 4. If this setting is set to "0", there will be minimal
logging and with setting "4" there will be maximum amount of logging.

Usable values in production use are 0, which logs only file transfers
and 1, which logs file transfers and device inquiries.

WARNING! Full verbose logging (4) should be used only for debugging
purposes, since it creates a lot of logs and if you are logging to
flash, it may fill it up.
```

5. Inquiry logging [Yes]

```
The default value for this option is "Yes".

Toggles logging of inquiry messages on/off.

WARNING! Inquiry messages may fill up diskspace quickly or use all
available network bandwidth.
```

6. Block list save delay [0]

```
The default value for this option is "0".

Determines how often (in seconds) a dump file is updated. Using a dump file
allows the blocklist to be saved in case of power failure of Access Server.
"0" disables this feature. We recommend to use a rather big value, for
example 15min = 900s.

WARNING: Using a small value here can physically burn the flash memory
over time.
```

7. Block list file name [/var/lib/obexsender/blocklist.dump]

```
The default value for this option is "/var/lib/obexsender/blocklist.dump".

You can choose to save the information about already served devices,
so you can form a so-called "block list". If this ignore list is
saved in flash memory, it will be preserved even if Access Server is
rebooted. This basically ensures that remote devices do not receive
the same content even if Access Server is rebooted.
```

8. Flush block list [/sbin/service obexsender flushblock]

```
Remove all entries in the running block list. Use this if you need
to remove information about already served devices, for example
when you need to ensure that a new content specified with a time
directive gets sent to devices already served.
```

9. View log [-]

```
This link allows you to view ObexSender log file if it exists.
By default a summary of the logged events is displayed.
Detailed information is available by clicking the date links.
```

B.4.2.2.1. Delete log (confirm)

```
This link will delete the current log file after confirmation.
```

1. Delete log now! [/bin/false]

   ```
   Delete ObexSender log file immediately!

   WARNING: There is no confirmation for this!
   ```

B.4.3. wpkgd settings

```
This submenu contains settings for wpkgd application.
```

1. wpkgd's autoinstall directory [@]

   ```
   The default value for this option is "@".

   wpkgd will automatically check this directory for wpk files containing
   software update packets.

   Special meta character, "@", means Obexserver's root directory. Use it if
   you want to allow updates via Bluetooth ObjP or FTP profiles.
   Use empty to disable autoinstall.
   ```

2. Password for autoinstall packages []

   ```
   The default value for this option is empty.

   This is an optional password to authenticate wpk autoinstall packets (wpk
   packets sent to the autoinstall directory). The password is shown encrypted
   here, if set.

   To change the password, clear the field, enter a new password and click Save.

   Please note that the new password is shown in plain text only right after
   you have saved it. Later it is only shown encrypted, and there is no way
   to decrypt it. You must either remember it or change it again to
   something you do remember.

   Use "-" do disable the password.

   The password must match the authentication parameter in the "wpkg.pif"
   file in the wpk packet. Otherwise the packet is not processed.

   Syntax in the "wpkg.pif" file:
   %wpkg-auth: auth
   ```

3. Delete processed autoinstall packages [Yes]

   ```
   The default value for this option is "Yes".

   If this option is set to Yes, the wpk autoinstall packets are deleted
   after they have been processed.
   ```

4. Process hotplug packages [Yes]

   ```
   The default value for this option is "Yes".
   ```

```
If this option is set to Yes, wpk packets are automatically processed
from USB memory sticks or Compact Flash memory cards when they are
plugged into Access Server.
```

5. Password for hotplug packages []

```
The default value for this option is empty.

This is an optional password to authenticate wpk installation packets
automatically run from USB memory dongles or Compact Flash memory cards.
The password is shown encrypted here, if set.

To change the password, clear the field, enter a new password and click Save.

Please note that the new password is shown in plain text only right after
you have saved it. Later it is only shown encrypted, and there is no way
to decrypt it. You must either remember it or change it again to
something you do remember.

Use "-" to disable the password.

The password must match the authentication parameter in the "wpkg.pif"
file in the wpk packet. Otherwise the packet is not processed.

Syntax in the "wpkg.pif" file:
%wpkg-auth: auth
```

6. Delete processed hotplug packages [No]

```
The default value for this option is "No".

If this option is set to Yes, the wpk packets are deleted
after they have been processed.
```

7. Extra parameters for wpkgd []

```
The default value for this option is empty.

Optional extra command line parameters for wpkgd.

Please see wpkgd --help for detailed information on the options.
```

8. Keep device mounted after wpk check [No]

```
The default value for this option is "No".

Keep the device mounted after wpk check.
```

9. Mountpoint for device [/mnt/disk]

```
The default value for this option is "/mnt/disk".

A mountpoint for a device.
```

B.4.3.1. Boot time reflash

```
Boot time reflashing options.
```

1. Check boot time reflashing status [/usr/sbin/dataflasher --get-bootflash]

   ```
   Output current status of boot time reflashing.
   ```

2. Enable boot time reflashing [/usr/sbin/dataflasher --enable-bootflash]

   ```
   Enable boot time reflashing from USB storage.
   ```

3. Disable boot time reflashing [/usr/sbin/dataflasher --disable-bootflash]

   ```
   Disable boot time reflashing from USB storage.
   ```

B.5. iWRAP settings

```
This submenu contains all iWRAP related settings.
```

1. iWRAP password [buffy]

   ```
   The default value for this option is "buffy".

   This password is required to be entered before entering any commands when
   communicating with iWRAP.

   To change the password, clear the field, enter a new password and click Save.
   Saving an empty field keeps the old password.

   Please note that the new password is shown in plain text only right after
   you have saved it. Later it is only shown encrypted, and there is no way
   to decrypt it. You must either remember it or change it again to
   something you do remember.

   Use "-" to disable iWRAP password.
   ```

2. Allow local clients without password [Yes]

   ```
   The default value for this option is "Yes".

   If this setting is "Yes", iWRAP password is requested only from remote
   clients, not from local clients (127.0.0.1).
   ```

3. Friendly name [W$S_$p]

   ```
   The default value for this option is "W$S_$p".

   The name shown when this device is found when inquired about by other
   Bluetooth devices. The following meta tags are available:

   $S : Hardware serial number, all ten digits
   $s : Hardware serial number, last three digits
   $P : Server port
   $p : Server port, last digit
   $H : Fully Qualified Domain Name (FQDN)
   $h : hostname
   $$ : $

   For example, "Server_$p" would set the Bluetooth friendly name as
   ```

```
"Server_1" for first baseband, "Server_2" for second baseband and
"Server_3" for third baseband.
```

4. Connectable and discoverable mode [3]

```
The default value for this option is "3".

This setting specifies whether this device is connectable and/or
discoverable by other Bluetooth devices.

When a device is connectable, other Bluetooth devices can open a Bluetooth
connection to it. Before opening a connection, the calling device must know
the Bluetooth address of the device it is connecting to. The Bluetooth
addresses can be found by making an inquiry. When a device is discoverable,
it shows up in inquiries. Possible values for all combinations of these
settings are:

0 : Not connectable, not discoverable
1 : Not connectable, discoverable
2 : Connectable, not discoverable
3 : Connectable and discoverable
```

5. Master/slave role switch policy [1]

```
The default value for this option is "1".

This setting specifies how a local Bluetooth device should decide
its role. When a Bluetooth device connects to another Bluetooth device,
it is a master by default and the answering device is the slave. When the
connection is being built, a role switch can be made. Normally,
access point devices need to be the master, and therefore they
require a master-slave switch when a new device is connecting.
This is also how Access Server is configured by default. Otherwise
Access server could not host the maximum number of slaves (7).
Other possible combinations are:

0 : Allow switch when calling, do not request it when answering
1 : Allow switch when calling, request it when answering
2 : Do not allow switch when calling, request it when answering

If you have problems connecting to Access Server, it might be
because your client device does not support the master/slave switch.
In this case you should change this setting to 0.
```

6. Default PIN code []

```
The default value for this option is empty.

This PIN code is used when establishing connections. Up to 16 characters
can be used.

If there is no default PIN code set, Access Server does not require
a PIN code when establishing connections.

However, if there is no default PIN code set, but the other device
requests a PIN code, "0000" is replied.
```

7. Power save mode and parameters [4]

```
The default value for this option is "4".

The power save mode used by default for all connections. Possible settings
are:

0 : Active
1 : Park: Round-robin
2 : Park: Idle
3 : Sniff: All
4 : Sniff: Idle

"Active" means that no power saving is in use.
"Sniff: All" means that the connections are kept in sniff mode always.
"Sniff: Idle" means that a connection is switched to sniff mode after
it has not transmitted data for some time (2 seconds by default).
When data transmission resumes, switch to active mode is made.

Park modes are generally not useful. See User's and Developer's Guide
and Bluetooth specification for more information.
```

8. Use literal replies in SDP [Yes]

```
The default value for this option is "Yes".

If enabled, some SDP result codes will have literal values instead of
numeric values.
```

9. Optional command line parameters []

```
The default value for this option is empty.

Optional extra command line startup parameters for the iWRAP servers.
```

10. Edit startup script [/etc/bluetooth.conf]

```
Opens iWRAP configuration file (/etc/bluetooth.conf) for editing.

You can add extra iWRAP commands to that file. iWRAP servers process
the file each time they start. See the User's and Developer's Guide
for iWRAP command reference.

For example, a unique friendly name for each baseband can be added by
using the following lines:

10101 SET BLUETOOTH NAME Foobar
10102 SET BLUETOOTH NAME Barfoo
10103 SET BLUETOOTH NAME Buffy!
```

11. Restart iWRAP [/sbin/service bluetooth restart]

```
iWRAP service must be restarted to activate new settings.
```

B.5.1. Bluetooth profiles

This is a submenu for configuring the supported Bluetooth profiles.

1. Enable Device ID profile [Yes]

   The default value for this option is "Yes".

   Whether or not Device ID profile is enabled.

2. Enable LAN access profile [No]

   The default value for this option is "No".

   Whether or not the LAN Access Profile is enabled.

3. Enable PAN user profile [No]

   The default value for this option is "No".

   Whether or not the PAN User Profile is enabled.

4. Enable PAN generic networking profile [No]

   The default value for this option is "No".

   Whether or not the PAN Generic Networking Profile is enabled.

5. Enable PAN network access point profile [No]

   The default value for this option is "No".

   Whether or not the PAN Network Access Point Profile is enabled.

6. Enable object push profile [Yes]

   The default value for this option is "Yes".

   Whether or not the Object Push Profile is enabled.

7. Enable file transfer profile [Yes]

   The default value for this option is "Yes".

   Whether or not the File Transfer Profile is enabled.

B.5.1.1. LAN access profile settings

This submenu contains LAN Access Profile settings.

1. Login name and password []

   The default value for this option is empty.

   The login name and password required from LAN access clients. Must be entered as a single string, separated with a space. For example: guest buffy

   If empty, no login is required.

2. Service channel [4]

   The default value for this option is "4".

```
Service channel for LAN access profile.
```

3. Service name (shown in SDP) [LAN Access]

```
The default value for this option is "LAN Access".

The name of the LAN Access Profile service shown in the Service Discovery.
```

4. Defaultroute modification policy [0]

```
The default value for this option is "0".

How the LAN Access Profile should modify the defaultroute in routing tables:

0: Do not alter defaultroute
1: When acting as a LAP client, set defaultroute according to the LAP server
2: When acting as a LAP server, set defaultroute according to the LAP client
3: Set defaultroute according to the LAP server/client connected
```

5. First IP for LAP clients [192.168.160.0]

```
The default value for this option is "192.168.160.0".

This defines the C-class of IP addresses to be used in point-to-point
connections between Access Server and LAP clients.

Full C-class is required: use "x.y.z.0".
```

B.5.1.2. PAN user profile settings

```
This submenu contains Personal Area Network User Profile settings.
```

1. Service name (shown in SDP) [PAN User]

```
The default value for this option is "PAN User".

The name of the PAN User Profile service shown in the Service Discovery.
```

2. Enable zeroconf when calling [No]

```
The default value for this option is "No".

Enable ZeroConf protocol for outgoing PANU connections.
```

3. Enable zeroconf when answering [No]

```
The default value for this option is "No".

Enable ZeroConf protocol for incoming PANU connections.
```

B.5.1.3. PAN generic networking profile settings

```
This submenu contains Personal Area Network Generic Networking Profile
settings.
```

1. Service name (shown in SDP) [Generic Networking]

```
The default value for this option is "Generic Networking".

The name of the PAN Generic Networking Profile service shown in
the Service Discovery.
```

2. Use dynamic network configuration for local IP address [No]

   ```
   The default value for this option is "No".

   Whether or not DHCP is used for configuring the local IP Address. Enable only
   if you are connecting this PAN-GN to another PAN-GN that will provide the IP
   configuration.
   ```

3. Local GN interface IP address [192.168.161.1]

   ```
   The default value for this option is "192.168.161.1".

   The IP address for the local GN interface.
   ```

4. Local GN interface subnet mask [255.255.255.0]

   ```
   The default value for this option is "255.255.255.0".

   The netmask for the local GN interface.
   ```

5. Start DHCP server for remote users [Yes]

   ```
   The default value for this option is "Yes".

   Whether or not this device should start DHCP for remote devices connecting
   to this PAN-GN. Disabled if "Use dynamic network configuration for local IP
   address" is used.
   ```

6. First IP for lease block [192.168.161.2]

   ```
   The default value for this option is "192.168.161.2".

   First IP address of the lease block.
   ```

7. Last IP for lease block [192.168.161.254]

   ```
   The default value for this option is "192.168.161.254".

   Last IP address of the lease block.
   ```

8. Subnet of lease block [255.255.255.0]

   ```
   The default value for this option is "255.255.255.0".

   Subnet mask of the lease block.
   ```

9. Lease time [86400]

   ```
   The default value for this option is "86400".

   Lease time in seconds.
   ```

B.5.1.4. PAN network access point profile settings

```
This submenu contains Personal Area Network Network Access Point Profile
settings.
```

1. Service name (shown in SDP) [Network Access]

   ```
   The default value for this option is "Network Access".

   The name of the Bluetooth PAN Network Access Point Profile service
   ```

shown in the Service Discovery.

B.5.1.5. Connection forwarding

Connection forwarding allows you to configure Access Server to automatically receive and forward Bluetooth connections to other applications or TCP socket.

This submenu contains all the settings related to connection forwarding capability.

1. #1 Command [0.0.0.0:0]

   The default value for this option is "0.0.0.0:0".

   This is the application or TCP/IP address and port where the connection is forwarded. For L2CAP connections, use the following format: "L2CAP:host:port".

   Example:

   192.168.42.1:5001
   /usr/local/bin/myapp

2. #1 Service UUID [SERIALPORT]

   The default value for this option is "SERIALPORT".

   This configures the Bluetooth profile that is used for connection forwarding. For L2CAP connections, use the following format: "L2CAP:UUID".

   See User's and Developers Guide for supported UUIDs.

3. #1 Service channel [5]

   The default value for this option is "5".

   RFCOMM channel or L2CAP psm for the Bluetooth profile configured in the "Service UUID" configuration.

4. #1 Service name (shown in SDP) [Serial Port]

   The default value for this option is "Serial Port".

   Name shown in Service Discovery.

B.5.1.6. Object push profile settings

This submenu contains Object Push Profile settings.

1. Service channel [3]

   The default value for this option is "3".

   Service channel for Object Push Profile.

2. Service name (shown in SDP) [Object Push]

   The default value for this option is "Object Push".

   The name of the Object Push Profile service shown in the Service Discovery.

3. Root directory [/tmp/obex]

   ```
   The default value for this option is "/tmp/obex".

   Root directory for obexserver application.

   The files received with Object Push Profile and File Transfer Profile are
   saved into this directory.

   Note: "/tmp/obex" is in RAM filesystem and will be erased during reboot.
   ```

4. Optional parameters for server [--bdaddr $b --prefix $b-$P-]

   ```
   The default value for this option is "--bdaddr $b --prefix $b-$P-".

   Optional parameters for obexserver application. See "obexserver --help"
   or User's and Developer's Guide for a list of parameters.
   ```

B.5.1.7. File transfer profile settings

```
This submenu contains File Transfer Profile settings.
```

1. Service channel [3]

   ```
   The default value for this option is "3".

   Service channel for File Transfer Profile.
   ```

2. Service name (shown in SDP) [File Transfer]

   ```
   The default value for this option is "File Transfer".

   The name of the File Transfer Profile shown in the Service Discovery.
   ```

3. Root directory [/tmp/obex]

   ```
   The default value for this option is "/tmp/obex".

   Root directory for obexserver application.

   The files received with Object Push Profile and File Transfer Profile are
   saved into this directory.

   Note: "/tmp/obex" is in RAM filesystem and will be erased during reboot.
   ```

4. Optional parameters for server [--bdaddr $b --prefix $b-$P-]

   ```
   The default value for this option is "--bdaddr $b --prefix $b-$P-".

   Optional parameters for obexserver application. See "obexserver --help"
   or User's and Developer's Guide for a list of parameters.
   ```

B.5.1.8. Serial port profile settings

```
This submenu contains the Bluetooth Serial Port Profile settings.

The profile itself is enabled and disabled by switching "serialport"
application "on" or "off" from the menu:
Setup -> Applications -> Default startup applications.
```

1. Act as the calling device [No]

```
The default value for this option is "No".

Whether this device should act as the calling device (DevA) or the answering
device (DevB).
```

2. Device [/dev/ttyAT1]

```
The default value for this option is "/dev/ttyAT1".

Device to which Bluetooth Serial Port Profile is connected.

/dev/ttyUSB0 for USB modem
/dev/ttyUSB4 for USB modem (Sierra Wireless Compass(tm) 885)
/dev/ttyACM0 for USB modem (Falcom Samba 75)
/dev/ttyAT1 for user uart (Access Server only)
/dev/ttyS0 for serial port device in Compact Flash (Access Server only)
```

3. BPS rate [115200]

```
The default value for this option is "115200".

The bits-per-second rate of the connection. Possible values are:
300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, and 460800.
```

4. Data bits [8]

```
The default value for this option is "8".

The number of data bits in the connection. Possible values are:
5, 6, 7, and 8.
```

5. Parity [0]

```
The default value for this option is "0".

The parity bit setting of the connection. Possible values are:

0: No Parity (default)
1: Odd Parity
2: Even Parity
```

6. Stop bits [1]

```
The default value for this option is "1".

The number of stop bits in the connection. Possible values are 1 and 2.
```

7. Hardware flow control (RTS/CTS) [Yes]

```
The default value for this option is "Yes".

Whether or not the hardware flow control is used.
```

8. Software flow control (XON/XOFF) [No]

```
The default value for this option is "No".

Whether or not the software flow control is used.
```

9. Bluetooth address of the remote device [00:07:80:80:bf:01]

   ```
   The default value for this option is "00:07:80:80:bf:01".

   The Bluetooth address of the device to be contacted. If the local device
   is configured as DevA, this is the DevB it tries to connect.
   ```

10. Service channel [2]

    ```
    The default value for this option is "2".

    In DevA (call) mode:   The Bluetooth RFCOMM channel of the remote device.

    In DevB (answer) mode: The Bluetooth RFCOMM channel of the local device.
    ```

11. Service name (shown in SDP) [Serial Port]

    ```
    The default value for this option is "Serial Port".

    The name of the Bluetooth Serial Port Profile service shown in the Service
    Discovery.
    ```

12. Optional command line parameters []

    ```
    The default value for this option is empty.

    Optional extra parameters for the Access Server Serial Port profile
    application. Currently the supported parameters are:

    --msc            Enables transmitting of DCD/DSR Modem Status Control signals.
    --nobuffer       Discard data if no Bluetooth connection, do not buffer it.
    ```

B.6. Advanced settings

```
This submenu contains advanced settings of Access Server.
```

1. System startup script [/etc/rc.d/rc.local]

   ```
   This is the last initialization script executed at system startup.

   By default, the script /etc/rc.d/rc.local just turns off all LEDs to
   indicate the startup has finished. If you want to initialize something
   automatically at every boot, or start up your own applications,
   you should add the required commands to this file.

   Remember to start your programs to the background. Example:
   /usr/local/bin/myapp &

   If you do not start the programs to the background, you will not able
   to access the management console using a serial cable.
   ```

2. Default user profile [/etc/profile]

   ```
   Edit the file containing the default user profile settings.
   ```

3. Setup access [/etc/setup.conf]

   ```
   The "/etc/setup.conf" file can be used to give different access rights to
   different users of the WWW Setup.
   ```

```
The file consist of lines in following format:
example.tag +user1 +user2 -user3 -user4

This will allow (+) access to tag "example.tag" for "user1" and "user2"
and denies (-) access from "user3" and "user4". You can find the tags
from the output of
Setup -> Advanced -> System Information -> Collect info for support request

For example, the tag of this setting is advanced.setupconf. If you
have created another user "guest" in /etc/httpd.conf that can access
"/setup", you can deny that user from changing the Setup access settings
with the following line in this file:

advanced.setupconf -guest
```

4. Edit other configuration files []

   ```
   From this menu, you can edit any files located in Access Server file system.
   You can for example create "/var/spool/cron/crontabs/root" file for
   configuring the cron daemon.
   ```

5. Browse /tmp/obex files [/tmp/obex]

   ```
   Browse files stored in /tmp/obex directory.
   ```

6. Browse all files []

   ```
   Browse all files stored in Access Server.
   ```

7. Find other Access Servers [/usr/sbin/finder]

   ```
   Find other Access Servers in the network.
   ```

8. Upload a software update [/tmp/obex]

   ```
   Upload a software update file (*.wpk).

   Access Server supports a special management packet format (wpk), which
   can be used to update Access Server software components or to install
   custom software and configuration files. Please consult User's and
   Developer's Guide for more information.
   ```

B.6.1. Bluetooth commands

```
This submenu contains advanced Bluetooth commands.
```

1. Inquiry for Bluetooth devices [/usr/bin/btcli inquiry]

   ```
   This command runs a standard inquiry command and lists found devices.
   See User's and Developer's Guide for more information about inquiry.
   ```

2. Set Bluetooth radios to class 1 [/usr/sbin/btclass 1]

   ```
   This commands sets Bluetooth radios to class 1 power levels, max. +20dBm.
   ```

3. Set Bluetooth radios to class 2 [/usr/sbin/btclass 2]

   ```
   This commands sets Bluetooth radios to class 2 power levels, max. +4dBm.
   ```

4. Set Bluetooth radios to class 3 [/usr/sbin/btclass 3]

   ```
   This commands sets Bluetooth radios to class 3 power levels, max. 0dBm.
   ```

B.6.2. System information

```
This submenu contains tools to retrieve system status information.
```

1. Hardware information

   ```
   Displays hardware and software identification information (output of
   command "wrapid").
   ```

2. List installed software components [/usr/bin/dpkg -l]

   ```
   Lists currently installed software components and their version numbers.
   ```

3. List running processes [/bin/ps ww]

   ```
   Lists running processes (output of command "ps").
   ```

4. List memory status [/usr/bin/free]

   ```
   Lists memory status (output of command "free").
   ```

5. List free disk space [/bin/df -h]

   ```
   Lists free disk space (output of command "df -h").
   ```

6. Show syslog file [/var/log/messages]

   ```
   Shows syslog file.
   ```

7. Show boot log file [/var/log/dmesg]

   ```
   Shows boot log.
   ```

8. Collect info for support request [/usr/sbin/supportinfo]

   ```
   This page contains collectively all the system status and configuration
   information.

   Include this information when sending a support request to
   support@bluegiga.com

   WARNING: All classified information, like passwords, should be automatically
   excluded. It is still recommended to manually check that all such information
   is really removed.
   ```

B.6.3. Reboot system (confirm)

```
Reboot Access Server. Confirmation will be asked.
```

1. Reboot now! [/sbin/reboot]

   ```
   Reboot Access Server immediately!

   WARNING: There is no confirmation for this!
   ```

B.7. Summary of Setup Options

```
Security settings
    Root password                        [buffy]
    Setup password                       [buffy]
    iWRAP password                       [buffy]
    Allow local clients without password  [Yes]
    Bluetooth PIN code                   []
```

```
    wpkgd autoinstall password            []
    wpkgd hotplug password                []

Generic settings
    Root password                         [buffy]
    Description of this unit              [@A #@S]
    Use local syslog service              [Yes]
    Syslog file name                      [/var/log/messages]
    Size of syslog file                   [63]
    Number of rotated syslog files        [3]
    IP address of the remote syslog server          [192.168.42.1]
    Swap to NFS server                    [No]
    Hostname and directory for NFS swap   [swap.localdomain:/var/swap]
    NFS swap size in megabytes            [64]
    System clock tick                     [10000]
    System clock frequency                [0]

Network settings
    Hostname of the unit                  [wrap]
    Domain of the unit                    [localdomain]
    Default interface settings
        Use dynamic network configuration          [Yes]
        IP address                        [192.168.42.3]
        Subnet mask                       [255.255.255.0]
        IP address of the default gateway          [192.168.42.254]
        List of name server IPs           [192.168.42.1 192.168.42.2]
        DHCP server settings
            First IP for lease block      [192.168.42.4]
            Last IP for lease block       [192.168.42.253]
            Subnet of lease block         [255.255.255.0]
            Lease time                    [86400]
    Enable ethernet cable interface       [Yes]
    Ethernet cable settings
        Assign to default interface       [Yes]
        Use dynamic network configuration          [Yes]
        IP address                        [192.168.43.3]
        Subnet mask                       [255.255.255.0]
    Time server (NTP)                     []
    Time server (rdate)                   []
    Update current time now by NTP        [/sbin/service ntpd sync]
    Zeroconf interface                    [nap]
    DHCP client extra parameters          []
    Enable modem interface                [No]
    Modem settings
        Device                            [/dev/ttyUSB0]
        Force connection open             [No]
        IP address used in force check    [194.100.31.45]
        Username                          [blue]
        Password                          [giga]
        Extra parameters for pppd         []
        Edit pppd options file            [/etc/ppp/peers/gprs]
        Edit pppd connect script file     [/etc/ppp/peers/gprs.connect]
        Re-establish modem connection     [/sbin/service network restart]
```

```
        Collect modem diagnostics         [/usr/sbin/supportinfo --modem]
    Enable Wi-Fi interface                [No]
    Wi-Fi settings
        Act as a Wi-Fi Access Point       [No]
        ESSID                             []
        Nickname                          []
        WEP encryption key                []
        Extra commands for Access Point mode        [/etc/sysconfig/ifup-wlan0]
        Assign to default interface       [No]
        Use dynamic network configuration           [Yes]
        IP address                        [192.168.44.3]
        Subnet mask                       [255.255.255.0]
        Collect Wi-Fi diagnostics         [/usr/sbin/supportinfo --wifi]
        Use WPA encryption                [No]
        Edit WPA configuration file       [/etc/wpa_supplicant.conf]

Applications
    Default startup applications          []
    Connector settings
        Delay between calls               [20]
        Logfile name                      [-]
        Register to watchdog daemon       [Yes]
        Verbosity level                   [0]
        Edit configuration file           [/etc/connector.conf]
        #1 Bdaddr                         [-]
        #1 Channel                        [-]
        #1 Command                        [-]
    ObexSender settings
        Bluetooth friendly name           [W$S_$p]
        Minimum RSSI value before sending           [-80]
        Whitelist RSSI limit              [0]
        Require pairing                   [No]
        Timeouts and delays
            Delay between inquiries       [10]
            If previous was ok, timeout before sending again  [36000]
            If previous was fail, timeout before sending again          [86400]
            Delay between retrying call   [120]
            Delay after scanning          [5]
            Tester delay                  [60]
            Pair expire timeout           [0]
        Log file
            Log file name                 [-]
            Log prefix                    [-]
            If sending was failure, log it too      [Yes]
            Verbosity level               [0]
            Inquiry logging               [Yes]
            Block list save delay         [0]
            Block list file name          [/var/lib/obexsender/blocklist.dump]
            Flush block list              [/sbin/service obexsender flushblock]
            View log                      [-]
            Delete log (confirm)
                Delete log now!           [/bin/false]
        Delete non-matching requests      [Yes]
```

```
        Register to watchdog daemon       [Yes]
        Edit configuration file           [/etc/obexsender.conf]
        Inquiry and calculate hash        [/usr/sbin/obexsender-hash --download]
        Restart ObexSender                [/sbin/service obexsender hardrestart]
        Base directory for content files  [/usr/local/obexsender/files/]
        Upload new content file           [/usr/local/obexsender/files/]
        Browse content files              [/usr/local/obexsender/files/]
    wpkgd settings
        wpkgd's autoinstall directory     [@]
        Password for autoinstall packages           []
        Delete processed autoinstall packages       [Yes]
        Process hotplug packages          [Yes]
        Password for hotplug packages     []
        Delete processed hotplug packages           [No]
        Extra parameters for wpkgd        []
        Keep device mounted after wpk check         [No]
        Mountpoint for device             [/mnt/disk]
        Boot time reflash
            Check boot time reflashing status       [/usr/sbin/dataflasher --get-bootflash]
            Enable boot time reflashing   [/usr/sbin/dataflasher --enable-bootflash]
            Disable boot time reflashing  [/usr/sbin/dataflasher --disable-bootflash]

iWRAP settings
    iWRAP password                        [buffy]
    Allow local clients without password  [Yes]
    Friendly name                         [W$S_$p]
    Connectable and discoverable mode     [3]
    Master/slave role switch policy       [1]
    Default PIN code                      []
    Power save mode and parameters        [4]
    Use literal replies in SDP            [Yes]
    Optional command line parameters      []
    Edit startup script                   [/etc/bluetooth.conf]
    Bluetooth profiles
        Enable Device ID profile          [Yes]
        Enable LAN access profile         [No]
        LAN access profile settings
            Login name and password       []
            Service channel               [4]
            Service name (shown in SDP)   [LAN Access]
            Defaultroute modification policy        [0]
            First IP for LAP clients      [192.168.160.0]
        Enable PAN user profile           [No]
        PAN user profile settings
            Service name (shown in SDP)   [PAN User]
            Enable zeroconf when calling  [No]
            Enable zeroconf when answering          [No]
        Enable PAN generic networking profile       [No]
        PAN generic networking profile settings
            Service name (shown in SDP)   [Generic Networking]
            Use dynamic network configuration for local IP address      [No]
            Local GN interface IP address           [192.168.161.1]
            Local GN interface subnet mask          [255.255.255.0]
```

```
            Start DHCP server for remote users      [Yes]
            First IP for lease block      [192.168.161.2]
            Last IP for lease block       [192.168.161.254]
            Subnet of lease block         [255.255.255.0]
            Lease time                    [86400]
        Enable PAN network access point profile     [No]
        PAN network access point profile settings
            Service name (shown in SDP)   [Network Access]
        Connection forwarding
            #1 Command                    [0.0.0.0:0]
            #1 Service UUID               [SERIALPORT]
            #1 Service channel            [5]
            #1 Service name (shown in SDP)          [Serial Port]
        Enable object push profile        [Yes]
        Object push profile settings
            Service channel               [3]
            Service name (shown in SDP)   [Object Push]
            Root directory                [/tmp/obex]
            Optional parameters for server          [--bdaddr $b --prefix $b-$P-]
        Enable file transfer profile      [Yes]
        File transfer profile settings
            Service channel               [3]
            Service name (shown in SDP)   [File Transfer]
            Root directory                [/tmp/obex]
            Optional parameters for server          [--bdaddr $b --prefix $b-$P-]
        Serial port profile settings
            Act as the calling device     [No]
            Device                        [/dev/ttyAT1]
            BPS rate                      [115200]
            Data bits                     [8]
            Parity                        [0]
            Stop bits                     [1]
            Hardware flow control (RTS/CTS)         [Yes]
            Software flow control (XON/XOFF)        [No]
            Bluetooth address of the remote device  [00:07:80:80:bf:01]
            Service channel               [2]
            Service name (shown in SDP)   [Serial Port]
            Optional command line parameters        []
    Restart iWRAP                         [/sbin/service bluetooth restart]

Advanced settings
    System startup script                 [/etc/rc.d/rc.local]
    Default user profile                  [/etc/profile]
    Setup access                          [/etc/setup.conf]
    Edit other configuration files        []
    Browse /tmp/obex files                [/tmp/obex]
    Browse all files                      []
    Find other Access Servers             [/usr/sbin/finder]
    Upload a software update              [/tmp/obex]
    Bluetooth commands
        Inquiry for Bluetooth devices     [/usr/bin/btcli inquiry]
        Set Bluetooth radios to class 1   [/usr/sbin/btclass 1]
        Set Bluetooth radios to class 2   [/usr/sbin/btclass 2]
```

```
    Set Bluetooth radios to class 3   [/usr/sbin/btclass 3]
System information
    Hardware information
    List installed software components          [/usr/bin/dpkg -l]
    List running processes            [/bin/ps ww]
    List memory status                [/usr/bin/free]
    List free disk space              [/bin/df -h]
    Show syslog file                  [/var/log/messages]
    Show boot log file                [/var/log/dmesg]
    Collect info for support request  [/usr/sbin/supportinfo]
Reboot system (confirm)
    Reboot now!                       [/sbin/reboot]
```

Appendix C. Available Software Packages

Package	Description	Installed by Default
a2dp	Bluegiga iWRAP A2DP services.	no
bash	GNU Projects Bourne Again SHell, interactive shell with Bourne shell syntax.	no
bbreset	Bluegiga kernel module for resetting Bluetooth basebands.	yes
bluetooth	Bluegiga iWRAP service.	yes
bluez-hcidump	Bluetooth packet analyzer.	no
bluez-libs	Bluetooth libraries needed by bluez-hcidump.	no
bridge-utils	Linux ethernet bridging utilities, needed to manage bridging for Bluetooth PAN profiles and Wi-Fi Access Point functionality.	yes
bstool	Bluegiga Bluetooth baseband control utilities including btclass command.	yes
btcli	Bluegiga iWRAP server command line interface utility.	yes
btlogger	Bluegiga example: a simple Bluetooth RFCOMM server.	no
btserver	Bluegiga example: an advanced iWRAP client.	no
busybox	Provides tens of general userland utilities.	yes
chkconfig	Bluegiga utilities: chkconfig and service commands.	yes
cifs-client	Mount helper utility for Linux CIFS VFS client.	no
connector	Bluegiga Connector, service which automatically opens and maintains connections to specified Bluetooth devices.	yes
curl	Command line tool for transferring files with URL syntax.	no
dataflasher	Bluegiga system update and bootloader configuration utility.	yes
dfu	Bluegiga Bluetooth baseband firmware upgrade tool.	yes
dosfstools	DOS filesystem utils.	no
dun	Bluegiga iWRAP service helper application.	no
ed	POSIX-compliant line editor.	yes
finder	Bluegiga utility finding other Access Servers and Access Points in the network.	yes

Package	Description	Installed by Default
forkserver	Bluegiga example: the simplest Bluetooth RFCOMM server.	no
ftp	FTP client application.	no
ftpd	Simple FTP server.	no
gdbserver	Remote Server for the GNU Debugger.	no
glibc	The GNU C library.	yes
glibc-devel		no
helloworld	Bluegiga example: extending setup application.	no
heyu	X10 automation.	no
hostapd	Utility programs for WPA and RSN authenticator.	no
hostap-utils	Utility program to break your CF Wi-Fi card.	no
iptables	Administration tool for the Linux kernel IP packet filter.	yes
iptables-extra		no
iptables-ipv6		no
kernel	Linux kernel.	yes
kernel-modules	Core kernel modules.	yes
kernel-modules-bluez	Linux kernel BlueZ modules.	no
kernel-modules-hostap	Linux kernel module providing support for Compact Flash Wi-Fi cards with Intersil Prism 2/2.5/3 chipset.	no
kernel-modules-modem	Linux kernel modules providing support for USB modems.	yes
kernel-modules-wifi	Wi-Fi drivers and firmwares	no
kitt	Bluegiga utility for controlling LEDs (and buzzer).	yes
led	Bluegiga kernel module accessing LEDs (and buzzer).	yes
ledtest	Bluegiga example: LED control.	no
libbghw	Bluegiga hardware library.	yes
libbgnet	Bluegiga socket, iWRAP and watchdog access libraries.	yes
libbgobex	Bluegiga iWRAP OBEX libraries.	yes
libnl	Library for applications dealing with netlink sockets.	no
libpcap	Provides portable framework for low-level network monitoring. Needed by tcpdump.	no
lighttpd	Secure, fast, compliant, flexible and small memory footprint http server.	no
lighttpd-mod-access		no

Package	Description	Installed by Default
lighttpd-mod-accesslog		no
lighttpd-mod-alias		no
lighttpd-mod-auth		no
lighttpd-mod-cgi		no
lighttpd-mod-compress		no
lighttpd-mod-evasive		no
lighttpd-mod-evhost		no
lighttpd-mod-expire		no
lighttpd-mod-extforward		no
lighttpd-mod-fastcgi		no
lighttpd-mod-flv-streaming		no
lighttpd-mod-proxy		no
lighttpd-mod-scgi		no
lighttpd-mod-secdownload		no
lighttpd-mod-setenv		no
lighttpd-mod-simple-vhost		no
lighttpd-mod-status		no
lighttpd-mod-userdir		no
lighttpd-mod-usertrack		no
lighttpd-mod-webdav		no
lottery	Bluegiga example: lottery extension for obexsender service.	no
m2n	Bluegiga example: machine-2-network (M2N) with syslog.	no
make	The Make.	no
makesms	Bluegiga example: generating outgoing messages for Bluegiga SMS gateway.	no
maradns	DNS server.	no
mg	Mg is a Public Domain EMACS style editor.	no
ncurses	Library for displaying and updating text on text-only terminals.	no
net-snmp	Suite of applications used to implement SNMP v1, SNMP v2c and SNMP v3 using both IPv4 and IPv6.	no

Package	Description	Installed by Default
nfs-utils	NFS server.	no
obexbrowser	Bluegiga iWRAP utility. A command line OBEX client interface.	no
obexget	Bluegiga iWRAP OBEX utilities: obexput and obexget commands for transfering files to/from remote devices with ObjP/FTP support.	yes
obexsender	Bluegiga proximity marketing service.	yes
obexsender-db	Bluegiga proximity marketing device database.	yes
obexserver	Bluegiga iWRAP service: ObjP and FTP server.	yes
openntpd	NTP (RFC-1305) client and server.	yes
openssh	OpenSSH suite; server and client utilities.	yes
openssl	Toolkit implementing SSL v2/v3, TLS v1 and general purpose cryptography library.	yes
openssl-progs		no
openvpn	An open source VPN daemon.	no
pcmciautils	A suite of userspace tools for PCMCIA support in the Linux 2.6 kernel.	yes
perl	A programming language.	no
ppp	Point-to-Point Protocol userland driver.	yes
readline	GNU Readline library, providing set of functions for use by applications that allow users to edit command lines as they are typed in.	yes
rootfiles	Bluegiga Access Server and Access Point filesystem skeleton.	yes
rsync	rsync is an open source utility that provides fast incremental file transfer.	no
rzsz	Provides X/Y/Zmodem file transfer tools.	no
sbc	SBC encoder and decoder.	no
screen	Screen is a full-screen window manager that multiplexes a physical terminal between several processes, typically interactive shells.	no
screen-utf8		no
serial	Bluegiga example: hello world using serial port.	yes
serialport	Bluegiga iWRAP service: SPP client/server.	yes
setup	Bluegiga Access Server and Access Point configuration utility and commands wrapid and supportinfo.	yes
setup-helloworld	Bluegiga example: extending setup application.	no
smsgw	Bluegiga SMS Gateway.	no

Package	Description	Installed by Default
sqlite	SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine.	no
strace	System call trace, i.e. a debugging tool.	no
sysfsutils	These are a set of utilites built upon sysfs, a new virtual filesystem in Linux kernel versions 2.5+ that exposes a systems device tree.	yes
tcpdump	Utility to monitor network traffic.	no
termcap	Basic system library needed to access the termcap database.	yes
toolchain	Native toolchain.	no
tremor	Ogg Vorbis decoder, also known as Tremor.	no
tzdata	Timezone.	yes
uartmode	Bluegiga Access Server user serial port control utility.	yes
watchdog	Bluegiga user level watchdog.	yes
wireless-tools	Package containing utilities to manage Wireless LAN specific parameters.	yes
wpa-supplicant	Utility programs for WPA and RSN supplicant.	no
wpkgd	Bluegiga software component management service.	yes
www	Bluegiga example: demonstration of WWW server capabilities.	no
zlib	General purpose compression library.	yes

Table C-1. List of Available Software Packages

Appendix D. Enabled Busybox Applets

Command	Description
addgroup	Add a group or add an user to a group.
adduser	Add an user.
adjtimex	Tune kernel clock.
ar	Create, modify, and extract from archives.
arp	Manipulate ARP cache.
arping	Send arp request to a neighbour host.
ash	The ash shell.
awk	Pattern scanning and processing language.
basename	Strip directory and suffix from filenames.
bunzip2	A block-sorting file decompressor.
bzcat	Decompresses files to stdout.
bzip2	A block-sorting file compressor.
cat	Concatenate files and print on the standard output.
chgrp	Change group ownership.
chmod	Change file access permissions.
chown	Change file owner and group.
chroot	Run command or interactive shell with special root directory.
clear	Clear the terminal screen.
cmp	Compare two files.
comm	Compare two sorted files line by line.
cp	Copy files and directories.
cpio	Copy files to and from archives.
crond	A daemon to execute scheduled commands. This server is configurable through the `/var/spool/cron/crontabs/root` file or the **crontab** command in the same way as any Linux crond.
crontab	Maintain crontab files for individual users.
cryptpw	Output crypted string.
cut	Remove sections from each line of files.
date	Print or set the system date and time.
dd	Convert and copy a file.
delgroup	Delete group from system or user from group.
deluser	Delete user from system.
df	Report file system disk space usage.
diff	Find differences between two files.
dirname	Strip non-directory suffix from file name.
dmesg	Print or control the kernel ring buffer.

Command	Description
dpkg	A medium-level package manager for (.deb) packages.
dpkg-deb	Debian package archive (.deb) manipulation tool.
du	Estimate file space usage.
dumpleases	Display DHCP leases granted by udhcpd.
echo	Display a line of text.
egrep	Print lines matching a pattern.
env	Print the current environment or run a program after setting up the specified environment.
expr	Evaluate expressions.
false	Do nothing, unsuccessfully.
fgrep	Print lines matching a pattern.
find	Search for files in a directory hierarchy.
free	Display amount of free and used memory in the system.
fuser	Identify processes using files or sockets.
getty	Open a tty, prompt for a login name, then invoke /bin/login.
grep	Print lines matching a pattern.
gunzip	Expand files.
gzip	Compress files.
halt	Stop the system.
head	Output the first part of files.
hexdump	Ascii, decimal, hexadecimal, octal dump.
hostid	Print the numeric identifier for the current host.
hostname	Show or set the system's host name.
httpd	Web server, which is described in detail in Section 3.9.6. Another Web server, `lighttpd`, is available as a separate software component.
hwclock	Query and set the hardware clock (RTC).
id	Print user identity.
ifconfig	Configure a network interface.
inetd	Internet services daemon. Notice that this server is disabled by default. Use the WWW interface of **setup** application or the **chkconfig inetd on** command to enable it.
init	Process control initialization.
insmod	Simple program to insert a module into the Linux kernel.
ip	Linux ipv4 protocol implementation.
ipaddr	Displays addresses and their properties, adds new addresses and deletes old ones.
iplink	Network device configuration.
iproute	Advanced ip routing and network device configuration tools..
iptunnel	Tunnel over IP.

Command	Description
kill	Terminate a process.
killall	Kill processes by name.
klogd	Kernel log daemon.
less	Opposite of more.
ln	Make links between files.
logger	A shell command interface to the syslog system log module.
login	Sign on.
losetup	Set up and control loop devices.
ls	List directory contents.
lsmod	Program to show the status of modules in the Linux kernel.
lzmacat	Uncompress to stdout.
md5sum	Compute and check md5 message digest.
mdev	System device manipulation tool.
microcom	Copy bytes for stdin to TTY and from TTY to stdout. Minimal TTY terminal.
mkdir	Make directories.
mknod	Make block or character special files.
mkswap	Set up a Linux swap area.
mktemp	Make temporary filename (unique).
modprobe	Program to add and remove modules from the Linux kernel.
more	View file or standard input one screenful at a time.
mount	Mount filesystems.
mv	Move (rename) files.
netstat	Display networking information.
nice	Run a program with modified scheduling priority.
nohup	Run a command immune to hangups, with output to a non-tty.
nslookup	Query Internet name servers interactively.
passwd	Update a user's authentication tokens(s).
patch	Apply a diff file to an original.
pidof	List PIDs of all processes with names that match one specified.
ping	Send icmp echo_request to network hosts.
ping6	Send icmp echo_request to network hosts.
poweroff	Stop the system.
printf	Format and print data.
ps	Report a snapshot of the current processes.
pwd	Print name of current/working directory.
rdate	Get the time via the network.
readlink	Display value of a symbolic link.
realpath	Return the canonicalized absolute pathname.

Command	Description
reboot	Reboot the system.
renice	Alter priority of running processes.
reset	Reset the screen.
resize	Resize the screen.
rm	Remove files or directories.
rmdir	Remove empty directories.
rmmod	Remove a module from the Linux kernel.
route	Edit the kernels routing tables.
sed	Stream editor for filtering and transforming text.
sendmail	Send an email.
seq	Print a sequence of numbers.
sh	Shell, the standard command language interpreter.
sha1sum	Compute and check sha1 message digest.
sleep	Delay for a specified amount of time.
sort	Sort lines of text files.
strings	Print the strings of printable characters in files.
stty	Change and print terminal line settings.
su	Run a shell with substitute user and group ids.
sulogin	Single user login.
swapoff	Stop swapping to file/device.
swapon	Start swapping to file/device.
sync	Flush file system buffers.
sysctl	Read/write system parameters.
syslogd	System logger.
tail	Output the last part of files.
tar	Create, extract, or list files from a tar file.
tcpsvd	Create TCP socket, bind it to ip:port and listen for incoming connection. Run PROG for each connection.
telnet	User interface to the telnet protocol.
telnetd	Telnet daemon.
test	check file types and compare values.
tftp	TFTP client.
tftpd	TFTP server.
time	Run a program with arguments specified. When command finishes, command's resource usage information is displayed.
top	Provide a view of process activity in real time.
touch	Change file timestamps.
tr	Translate or delete characters.

Command	Description
traceroute	Print the route packets trace to network host.
true	Do nothing, successfully.
tty	Print the file name of the terminal connected to standard input.
ttysize	Print dimension(s) of standard input's terminal, on error return 80x25.
udhcpc	DHCP client.
udhcpd	DHCP daemon for providing automatic network configuration for clients in the network.
udpsvd	Create UDP socket, bind it to ip:port and wait for incoming packets. Run PROG for each packet, redirecting all further packets with same peer ip:port to it.
umount	Unmount file systems.
uname	Print system information.
uniq	Report or omit repeated lines.
unlzma	Uncompress file.
unzip	List, test and extract compressed files in a zip archive.
uptime	Tell how long the system has been running.
usleep	Sleep some number of microseconds.
uudecode	Decode a binary file.
uuencode	Encode a binary file.
vconfig	Vlan (802.1q) configuration program.
vi	Screen-oriented (visual) display editor.
wc	Print the number of newlines, words, and bytes in files.
wget	The non-interactive network downloader.
which	Shows the full path of (shell) commands.
whoami	Print effective userid.
xargs	Build and execute command lines from standard input.
yes	Output a string repeatedly until killed.
zcat	Expand and concatenate data.
zcip	Manage a ZeroConf IPv4 link-local address.

Table D-1. List of Enabled Busybox Applets

Appendix E. Tested 3rd Party Peripherals

Name	Type	Software	Supported Functions/Modem Port	Notes
A-LINK WLAN54MB	Wi-Fi	kernel-modules-wifi	802.11b/g, WEP, WPA, WPA2	both internal and external antenna versions supported
Asus WL168G V2	Wi-Fi	kernel-modules-wifi	802.11b/g, AP, WEP, WPA, WPA2	FCC ID MSQWL167G
D-Link DWL-G122, H/W ver C1	Wi-Fi	kernel-modules-wifi	802.11b/g, AP, WEP, WPA, WPA2	FCC ID KA2WLG122C1
Edimax EW-7318MUg	Wi-Fi	kernel-modules-wifi	802.11b/g, AP, WEP, WPA, WPA2	
Linksys WUB54GC-EU	Wi-Fi	kernel-modules-wifi	802.11b/g, AP, WEP, WPA, WPA2	FCC ID Q87-WUSB54GC
Linksys WUSB200	Wi-Fi	kernel-modules-wifi	802.11b/g, AP, WEP, WPA, WPA2	FCC ID Q87-WUSB200, uses USB-B so always needs a cable.
Netwjork W541U	Wi-Fi	kernel-modules-wifi	802.11b/g, AP, WEP, WPA, WPA2	
Sunshine WLAN HWUG1	Wi-Fi	kernel-modules-wifi	802.11b/g, AP, WEP, WPA, WPA2	FCC ID NDD957318S607
Winxim WM802RTG	Wi-Fi	kernel-modules-wifi	802.11b/g, AP, WEP, WPA, WPA2	Only OEM version tested
Falcom Samba 75	GPRS	kernel-modules-modem	/dev/ttyACM0	In `gprs.connect`, requires ATD command in format "ATD*99***1#". If you need this device to work with Bluegiga SMS Gateway Server, please contact <`support@bluegiga.com`>
Huawei EG162G	GPRS	kernel-modules-modem	/dev/ttyUSB0	

Name	Type	Software	Supported Functions/Modem Port	Notes
Huawei E169	GPRS	kernel-modules-modem	/dev/ttyUSB0	Micro-SD reader not supported
Huawei E220	GPRS	kernel-modules-modem	/dev/ttyUSB0	Uses mini-USB-B so always needs a cable
Mobidata	GPRS	kernel-modules-modem	/dev/ttyUSB0	
Newolution Webbox	GPRS	built-in	/dev/ttyUSB0	Uses USB-B so always needs a cable. If you need this device to work with Bluegiga SMS Gateway Server, please contact <support@bluegiga.com>.
Sierra Wireless Compass 885	GPRS	kernel-modules-modem	/dev/ttyUSB4 (ttyUSB3 for SMS GW)	FCC ID N7NC885, micro-SD reader (support unconfirmed)
Teltonika ModemUSB/G10	GPRS	built-in	/dev/ttyUSB0	
Teltonika ModemUSB/H7.2, U3G15L	GPRS	kernel-modules-modem	/dev/ttyHS3	
Wavecom Q2686	GPRS	kernel-modules-modem	/dev/ttyACM0	Tested with evaluation kit. `gprs.connect` needs "ip" spelled upper case ("IP") and "ATD*99***1#"
Any vendor	Storage device	built-in	/dev/sda or /dev/sda1	If you find a USB storage device that is not working, please contact <support@bluegiga.com>

Table E-1. USB Peripherals Supported by Access Server and Access Point

Name	Type	Software	Supported Functions	Notes
AmbiCom WL1100C-CF	Wi-Fi	kernel-modules-hostap	802.11b, AP, WEP (WPA, WPA2 only as client)	

Name	Type	Software	Supported Functions	Notes
Canon K30225 (OEM)	Wi-Fi	kernel-modules-hostap	802.11b, AP, WEP (WPA, WPA2 only as client)	Only OEM version tested
D-Link DCF-660W	Wi-Fi	kernel-modules-hostap	802.11b, AP, WEP (WPA, WPA2 only as client)	
Anycom GS-320 Tri-Band GPRS CF Card	GPRS	built-in	Multislot class 10	Modem device /dev/ttyS0
Audiovox RTM 8000	GPRS	built-in	Multislot class 8	Modem device /dev/ttyS0, "same" HW as Fujitsu; Bluegiga SMS Gateway Server does not support this device.
Enfora GSM/GPRS Compact Flash Card (GSM 0110)	GPRS	built-in	Multislot class 8	Modem device /dev/ttyS0; Bluegiga SMS Gateway Server does not support this device.
Fujitsu Siemens Connect2Air 3GSM	GPRS	built-in	Multislot class 8	Modem device /dev/ttyS0, "same" HW as Audiovox
Pretec CompactGPS™	GPS	built-in		Serial port device /dev/ttyS0, 9600bps
Rikaline GPS-6021-X6	GPS	built-in		Serial port device /dev/ttyS0, 1200bps
Any vendor	Memory	built-in		If you find a card that does not work, please contact <support@bluegiga.com>

Table E-2. Compact Flash Cards Supported by Access Server

Name	Type	Notes
Buffalo WLI-USB-KB11	USB Wi-Fi	
D-Link DWA-140	USB Wi-Fi	
Edimax WE-7718UN	USB Wi-Fi	
TP-LINK TL-WN620G ver 1.2	USB Wi-Fi	
MTX-H25	USB GPRS	

Name	Type	Notes
Option Icon 225	USB GPRS	
Ambicom WL54-CF	CF Wi-Fi	No chipset support in kernel
Linksys WCF12	CF Wi-Fi	
Pretec 802.11g	CF Wi-Fi	
Pretec OC-WLBXX-A	CF Wi-Fi	
SMC EZ Connect SMC2642W V2 EU	CF Wi-Fi	

Table E-3. Peripherals Not Supported by Access Server or Access Point