IPDial

Stefan Gybas

COLLABORATORS			
	TITLE : IPDial		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Stefan Gybas	July 20, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

IPDial

1.1 IPDial 2.1 Documentation

```
        IPDial V2.1
        ===========

A dialer for SLIP/PPP connections with shell terminal program, released under
the GNU General Public License (see included file COPYING).


         Introduction       What's all this about?
         Features           What makes IPDial different from other dialers?
         License            GNU General Public License

         Installation       How to install IPDial on your machine
         Updating           Read this if you have used IPDial before
         Using IPDial       How can IPDial be started?

         Script commands    IPDial's own script language
         Common problems    Read this if you script does not work

         History            What has changed between the releases?
         Future             What might be implemented next?

         Acknowledgements   Thanks go to...
         Authors            Who wrote all this?


The authors give ABSOLUTELY NO warranty that the program described in this
documentation and the results produced by them are correct. The authors
cannot be held responsible for any damage resulting from the use of this
software.
```

1.2 Introduction to IPDial

```
IPDial is used to establish a SLIP, CSLIP or PPP connection to an ISP
(Internet service provider). This is done using a script file which consists
of commands like SEND "ATZ" or WAIT "CONNECT". After the connection is
```

established, IPDial usually logs in with your user account and activates SLIP/PPP on the remote machine. When all this is successfully done, IPDial exits and your SLIP/PPP device takes over the line.

When you want to disconnect, you can also use IPDial to make to modem physically hang up the line after SLIP/PPP has terminated.

Furthermore, IPDial has a built-in terminal mode which uses the current shell window. If something fails while IPDial is trying to establish a connection, you can just press Ctrl-F to enter the terminal mode and log in manually.

1.3 Features of IPDial

IPDial has several features that many other dialers lack:

- built-in shell terminal mode: You can always interrupt your dial script if something went wrong or enter a secret password without writing it into your dial script file.

- IPDial can read the serial configuration (device, unit, baud rate and handshake protocol) from a SANA configuration file. If you ever want to use a different device, unit or baud rate, just change it for your SANA device and IPDial will automatically use your new settings.

- own script language: Simple but powerful BASIC-like language. You can, for example, scan the input stream for dynamic IP addresses and store them in environment variables.

- very small: IPDial is less than 23 KB small and uses very little memory while running.

- sample scripts included: Just adapt one of the example scripts to your needs. You don’t have to write your own programs!

- IPDial is absolutely free of charge with full source code included (see the GNU General Public License for details).

1.4 Installing IPDial on your computer

To install IPDial on your machine just copy the executable IPDial to a directory in your command path. If you are using AmiTCP, AmiTCP:bin might be a good choice.

Next, adapt one of the supplied example login scripts (Login-SLIP.IPDial or Login-PPP.IPDial) and the hangup script Hangup.IPDial to your needs and copy them to your hard disk (AmiTCP:db might be a good place). For a list of script commands, see chapter Script commands.

1.5 Updating from IPDial 1.x and 2.0

IPDial 2.1 is based on IPDial 1.7, the last release from Jochen. The changes from IPDial 1.8/1.9 have not been included - if you really need the delay after DoIO() (as in IPDial 1.8/1.9) just recompile IPDial with a delay value set in IPDial.h.

You can use your old dial scripts but you might have to change them because IPDial 2.1 is not 100% compatible to IPDial 1.x and 2.0:

- the DELAY command now takes 2 arguments: seconds and milliseconds. If you haven't used decimals before, you don't have to change anything. Otherwise change e.g.

  DELAY 1.5 to DELAY 1 500

  DELAY 0.25 to DELAY 0 250

  and so on.

- the arguments of the SEND command are now separated by spaces (as the ECHO command alway did). If you have only been using quoted strings you don't have to change anything. Otherwise change e.g.

  SEND $MyUsername \r to SEND $MyUsername\r

  SEND a b c d to SEND abcd

  and so on.

- if you have used multiple arguments for one command, either put them in one paar of quotes or don't use quotes at all. Things like

  SYSTEM "Copy" "ENV:MYVAR ENVARC:MYVAR"

  won't work - use

  SYSTEM "Copy ENV:MYVAR ENVARC:MYVAR"

  of

  SYSTEM Copy ENV:MYVAR ENVARC:MYVAR

  instead.

- the SHOWPARAMS and SET commands have been removed in IPDial 2.1. Some options of the SET command (baud, protocol) have been moved to the DEVICE command. The other options are not needed IMHO. If you have use SET before, change

  DEVICE serial.device UNIT=0
  SET BAUD=38400 PROTOCOL=7WIRE [...]

  to

```
  DEVICE serial.device UNIT=0 BAUD=38400 7WIRE

  and remove all lines containing a SET or SHOWPARAMS command.

  I recommend using the new SANADEV command line option and removing the
  DEVICE command from your dial script. This way, you don't have to change
  your dial script, if your serial device changes.
```

1.6 Using IPDial

```
SYNOPSIS

    IPDial SCRIPT,DEVICE/K,PROTOCOL/K,UNIT/K/N,BAUD/K/N,SANADEV/K,
           TERMINAL/S,ECHO/S,VERBOSE/S,RAW/S


ARGUMENTS

    SCRIPT:    Script file to execute [S]
    DEVICE:    Device to use (default: "serial.device") [T]
    PROTOCOL:  Protocol to use: XONXOFF, 7WIRE (default) or NONE [T]
    UNIT:      Unit to use (default: 0) [T]
    BAUD:      Baud rate to use (default is taken from serial prefs) [T]
    SANADEV:   SANA config file to read serial config from [T,S]
    TERMINAL:  Run in terminal mode (i.e. don't execute script)
    ECHO:      Show modem replies [S]
               Use local echo mode [T]
    VERBOSE:   Show all operations the program performs [T,S]
    RAW:       Use character mode instead of line mode [T]

    If IPDial is called with TERMINAL switch given, SANADEV or (if SANADEV
    is not given) DEVICE, UNIT, BAUD and PROTOCOL are used to open the
    serial device. Once it is opened, the program acts like a very simple
    shell terminal program (terminal mode).

    If TERMINAL is omitted, the given SCRIPT file is read and the commands
    inside are executed line by line (script mode).

    [S]: argument is used in script mode, [T]: argument is used in terminal
    mode


EXAMPLES

    IPDial AmiTCP:db/Login-PPP.IPDial SANADEV=ppp0

    reads DEVICE, UNIT, BAUD and PROTOCOL from ENV:sana2/ppp0.config (the
    configuration file for PPP.device unit 0) and starts IPDial in script
    mode. All commands in AmiTCP:db/Login-PPP.IPDial are executed line by
    line until the end of file or an EXIT command is reached.


    IPDial DEVICE=artser.device BAUD=38400 TERMINAL RAW

    opens artser.device unit 0 (default) with 38400 baud and 7WIRE handshake
```

```
    (default) and acts like a simple shell terminal program (character mode).
    Each character entered is immediately sent to artser.device (the modem),
    while each received character is printed to the shell window.


CALLING IPDIAL FROM SHELL SCRIPTS

    IPDial is normally called from a shell script when you want to make a
    connection to your Internet provider. I recommend writing a small script
    called "DialSLIP" or "DialPPP" (e.g. in AmiTCP:bin) which contains the
    following lines (replace ppp0 with your SANA interface):

    AmiTCP:bin/IPDial AmiTCP:db/Login-PPP.IPDial SANADEV=ppp0
    If NOT WARN
      Execute AmiTCP:bin/StartNet
    EndIf

    To call your provider just enter "DialPPP" or "DialSLIP". To hangup your
    connection, create a second script called "StopPPP" or "StopSLIP" which
    contains the following lines (again replace ppp0 with your
    SANA interface):

    Execute AmiTCP:bin/StopNet
    AmiTCP:bin/IPDial AmiTCP:db/Hangup.IPDial SANADEV=ppp0


TERMINAL MODE

    If something went wrong with your dial script you can always enter
    terminal mode by pressing Ctrl-F and try to activate SLIP/PPP manually.
    When in terminal mode, you have several ways to leave:

    - Ctrl-C: exit IPDial with return code 5 (WARN)
    - Ctrl-F: exit IPDial with return code 0
    - Ctrl-\: continue script (if terminal mode was activated by a
      TERMINAL command in a dial script)

    If you failed to manually log in, just leave with Ctrl-C to tell your
    shell script not to  start AmiTCP. Otherwise use Ctrl-F to let your shell
    script know that everything is fine and AmiTCP with SLIP/PPP might be
    started.
```

1.7 IPDial's script language

```
- Any line of the script file may contain only one command. In general
  command arguments are parsed with ReadArgs(), thus they may look like CLI
  command line arguments: The characters "" may surround a string which
  contains blanks.

  Note, that ReadArgs() treats the character '*' as an escape sequence: Thus
  you have to write ECHO "**" if you want to print a single '*'.

- Empty lines or lines beginning with a semicolon are assumed to be comments
  and thus ignored.
```

```
- Lines may begin with a label, an alphanumeric word followed by a colon.
  Labels are ignored, except that they may be used as destinations for
  GOTO instructions. Anyone said BASIC? Yes, it is. :-)

- Labels are case-sensitive, commands are case-insensitive.

- This is the list of possible commands:

  DELAY SECS/A/N,MILLISECS/N
  DEVICE NAME/A,UNIT/K/N,BAUD/K/N,HANDSHAKE
  ECHO TEXT/A/F
  EXIT RETURNCODE/N
  GOTO LABEL/A
  ON STATUS GOTO LABELS/A/M
  SCAN FORMAT/A,GLOBAL/S,SAVE/S
  SEND TEXT/A/F
  SETVAR NAME/A,VALUE/A,GLOBAL/S,SAVE/S
  SYSTEM COMMAND/A/F
  TERMINAL EOF,NOECHO/S,RAW/S
  WAIT TIMEOUT/A/K/N,TEXT/A/M

See the scripts Login-SLIP.IPDial, Login-PPP.IPDial, T-Online.IPDIAL and
Hangup.IPDial as examples.
```

1.8 IPDial's script language: DELAY

```
SYNOPSIS

    DELAY SECS/A/N,MILLISECS/N

DESCRIPTION

    Delays the given number of seconds (plus milliseconds if given). While
    waiting, IPDial may be aborted with Ctrl-C (exit with RC 5) or Ctrl-F
    (enter terminal mode).

EXAMPLES

    DELAY 0 500    -- waits 0.5 seconds
    DELAY 2        -- waits 2 seconds
```

1.9 IPDial's script language: DEVICE

```
SYNOPSIS

    DEVICE NAME/A,UNIT/K/N,BAUD/K/N,HANDSHAKE

DESCRIPTION

    Opens the given device NAME, unit UNIT (default 0) at the given baud rate
    (default 9600). This device must be compatible to the serial.device.
    HANDSHAKE may be one of XONXOFF, RTSCTS (default), 7WIRE (equal to RTSCTS)
```

```
    or NONE.

    The DEVICE command should in general be the first command of each script
    unless you use the command line option SANADEV.

EXAMPLES

    DEVICE serial.device     -- opens serial.device unit 0 at 9600 baud with
                                RTSCTS handshake

    DEVICE bscisdn.device UNIT=1 BAUD=64000 -- opens bscisdn.device unit 1 at
                                               64000 baud with RTSCTS
```

1.10 IPDial's script language: ECHO

```
SYNOPSIS

    ECHO TEXT/A/F

DESCRIPTION

    This command will write the given line to stdout. Note that the text may
    contain patterns like \r (Carriage Return), \n (Line Feed), \\
    (Backslash) or \037 (octal digits, representing the character ASCII 31).

    Sequences like $VAR or ${VAR} will be replaced with the value of the
    environment variable VAR (local or global). If VAR doesn't exist, nothing
    is inserted. Use $$ to get the $ character itself. Please note that $VAR
    will only work, if the name VAR consists of alphanumeric characters and
    the name is separated with a non-alphanumeric character from the
    following characters. For example, $VAR+NAME means $VAR and not $VAR+NAME.
    On the other hand $VARNAME means $VARNAME and not $VAR.

    Note that ECHO does not write any Line Feeds or Carriage Returns unless
    you explicitly request it with the respective patterns.

EXAMPLES

    ECHO Hello, world!\n

    ECHO $USER "has logged in\n"

    ECHO The variable \"$$VAR\" contains ${VAR}\n
```

1.11 IPDial's script language: EXIT

```
SYNOPSIS

    EXIT RETURNCODE/N

DESCRIPTION
```

```
   Terminates the program, returns the given RETURNCODE (default 0).

EXAMPLES

   EXIT       -- exits IPDial with RC 0

   EXIT 5     -- exit IPDial with RC 5 (WARN)
```

1.12 IPDial's script language: GOTO

```
SYNOPSIS

    GOTO LABEL/A

DESCRIPTION

    Jumps to the given label and continues the script at that point.

EXAMPLES

    GOTO Logon     -- jumps to the label Logon (a line starting with
                      "Logon:") and executes the following commands.
                      Look at the ON command for a better example.
```

1.13 IPDial's script language: ON STATUS GOTO

```
SYNOPSIS

    ON STATUS GOTO LABELS/A/M

DESCRIPTION

    An ON command must follow a WAIT or SCAN command, because ON reads the
    value of the STATUS variable (set by WAIT and SCAN) and jumps to the first
    label if STATUS is -1, to the second label if STATUS is 0 and so on.

    You don't have to supply a label for each possible value of STATUS. If no
    lable is given, ON will suppress jumping and continue on the next line.

EXAMPLES

    DialAgain:
        [...]
        WAIT TIMEOUT=10 "Login:" "Busy"
        ON STATUS GOTO TimeOut Login Busy

    Login:
        [...]
        EXIT 0

    TimeOut:
        ECHO "Timeout happened, aborting.\n"
```

```
        EXIT 10

    Busy:
        ECHO "Remote busy, delaying...\n"
        DELAY 25
        ECHO "Trying again.\n"
        GOTO DialAgain
```

1.14 IPDial's script language: SCAN

```
SYNOPSIS

    SCAN FORMAT/A,GLOBAL/S,SAVE/S

DESCRIPTION

    The SCAN command is used to scan the buffer read by the last WAIT
    command for certain words. This may be used set environment
    variables with a part of the received text. The format string may
    contain the following patterns:

        %{WORD%} Ignores any characters until WORD is found in
                 the buffer. Use %% to insert the '%' character
                 into WORD.

        ' '      Ignores any number (including 0) and kind (blank,
                 tab, line feed, carriage return, form feed) of
                 white space characters.

     %[VAR%]%(SUFFIX%) Reads the next word from the buffer until
                 the first white space character and stores it into
                 the environment variable VAR.
                 The optional SUFFIX is a sequence of characters
                 to be removed from the end of the word.
                 Any other characters in the format string are simply
                 ignored.

    Usually environment variables are local (for IPDial and child processes
    only). Use the GLOBAL and SAVE (OS 3.0 only) keywords to store them in
    ENV: and ENVARC:, respectively. Note, that SAVE implies GLOBAL.

    Please note that you can use the SCAN command more than once on the same
    buffer.

    The SCAN command stores the number of created environment variables in
    the STATUS variable, which can later be used by the ON command. Note,
    that the value -1 is never stored in the STATUS variable, you have to
    use a dummy label in the ON command.

EXAMPLES

    SCAN "%{Your IP address is%} %[IPADDRESS%]%(.%)"

    scans the buffer for a sentence like
```

```
    Your IP address is 145.2.1.34.

    and stores the value 145.2.1.34 into the variable IPADDRESS. Note that
    the character '.' is removed.
```

1.15 IPDial's script language: SEND

```
SYNOPSIS

    SEND TEXT/A/F

DESCRIPTION

    This command sends the given strings to the serial device using
    DoIO(). These strings may contain the same patterns as described
    in the ECHO command.

EXAMPLES

    SEND "ATZ\n"

    SEND $PASSWORD\n
```

1.16 IPDial's script language: SETVAR

```
SYNOPSIS

    SETVAR NAME/A,VALUE/A,GLOBAL/S,SAVE/S

DESCRIPTION

    Sets environment variable NAME to VALUE. If you set the GLOBAL switch,
    your variable will be set in ENV: and not in the programs local
    environment. The SAVE switch forces copying to ENVARC: (OS 3.x only).
    Note, that SAVE implies GLOBAL.

EXAMPLES

    SETVAR USERNAME "Joe User" GLOBAL    -- sets the global variable USERNAME
                                            to "Joe User" (without quotes)
```

1.17 IPDial's script language: SYSTEM

```
SYNOPSIS

    SYSTEM COMMAND/A/F

DESCRIPTION

    Executes the command given by COMMAND. The arguments will be parsed like
```

```
    ECHO arguments (you may insert \ and $ patterns).

EXAMPLES

    SYSTEM Echo Hello!

    SYSTEM "C:List #?.IPDial >RAM:IPDial-Files"
```

1.18 IPDial's script language: TERMINAL

```
SYNOPSIS

    TERMINAL EOF,NOECHO/S,RAW/S

DESCRIPTION

    Enters terminal mode: What you enter at the keyboard will be sent to
    the serial device and likewise the program will display any input from
    the serial.device to you. The TERMINAL command will be finished, if
    you enter EOF (Ctrl-\).

    The NOECHO and EOF options can be used to enter a password, if you
    don't like to include it into your login file: NOECHO makes your input
    invisible and the EOF string terminates terminal mode as soon, as you
    type in the first character of the string.

    Usually you can send only complete lines to the modem, especially this
    means that you have all editing capabilities of the shell available.
    This is not the case in RAW mode: Every character you type will be sent
    immediately to the modem without any buffering or conversions. NOECHO
    mode implies RAW mode.

EXAMPLES

    WAIT TIMEOUT=10 "Password:"
    ON STATUS GOTO TimeOut
    ECHO "Enter login password: "
    TERMINAL EOF="\r" NOECHO
```

1.19 IPDial's script language: WAIT

```
SYNOPSIS

    WAIT TIMEOUT/A/K/N,TEXT/A/M

DESCRIPTION

    This command waits until either one of the given strings is read from
    the serial device or the number of seconds given by TIMEOUT has gone.

    A variable called STATUS indicates what happened: It contains either
    -1 for timeout or the number of the string that was read, beginning
```

```
    with 0. This variable may be used by the ON statement.

    WAIT arguments are parsed like ECHO arguments.

EXAMPLES

    WAIT TIMEOUT=10 "CONNECT" "BUSY" "NOCARRIER"
    ON STATUS GOTO TimeOut Connect Busy NoCarrier
```

1.20 Common problems with IPDial

- There might be problems with Executive (the task scheduler) if IPDial's priority is too low to get all characters from your serial device (see Executive's manual for further details). To prevent these problems either start IPDial with a priority above Executive's catch range (e.g. 3) or tell Executive to keep IPDial's priority above its schedule range.

- The following part of a dial script does not work:

```
  WAIT TIMEOUT=3 "IP address"
  ON STATUS GOTO TIMEOUT
  SCAN "%{Your IP address is%} %[IPADDRESS%]%(.%)" GLOBAL
```

  The SCAN command works on the buffer that was received until the previous WAIT command exited (either a string was found or timeout). You are trying to wait for "IP address" which is sent to you before the IP address, so it might not be in the received buffer.

  This seems to work in IPDial 1.x but that is not true: IPDial 2.0 is more optimized and faster than IPDial 1.x, so the WAIT command is left more quickly.

  To solve the above problem, wait for a string that is sent to you after the IP address. In some cases

```
  WAIT TIMEOUT=3 ".\n"  or  WAIT TIMEOUT=3 ".\r"
```

  might be a solution. If that does not work, wait for the whole line that is sent to you right before the IP address, e.g. use

```
  WAIT TIMEOUT=3 "SLIP ready. Your IP address is "
```

  . If that still does not work, use something like

```
  WAIT TIMEOUT=10 "a string that will never be sent to me"
```

  and hope that 10 seconds are enough to get your IP address.

1.21 Development of IPDial

```
Changes in IPDial 1.1 to 1.7

V 1.1  23.11.94  Initial version

V 1.2  27.02.95  Added terminal mode
                 Now using ReadArgs() for command line parsing.

V 1.3  09.03.95  Added environment variable support to "send"
                 command. Added unit support. Suggested by Quarvon
                 (Jürgen Lang)

V 1.4  21.04.95  Added "system" command. Suggested by Gutgolf
                 (Michael Bauer)

                 Added environment variable support to "echo" command.
                 Terminal mode now converts LF to CR/LF, so that modem
                 recognizes commands. (Let's hope, that will still
                 work for entering passwords.)

V 1.5  30.04.95  Added "scan" command.

V 1.6  26.06.95  "delay" command supporting ticks; ParseString()
                 supporting octal characters; "wait" command using
                 ParseString(). Suggested by Will Bow.

                 Fixed bug in SerialSend():
                 *.io_Device = *.io_Unit

V 1.7  21.07.95  Added NOECHO, RAW and EOF options to "terminal" command.
                 Suggested by Klaus Heinz

                 Added BAUD option to command line.

                 Added "setvar" command.


Changes in IPDial 2.0 (06.03.96)

+ new command line option SANADEV/K (reads serial device, unit, baud and
  handshake from SANA configuration file)
  IPDial.c/main(), IPDial.c/Usage(), added SanaConfig.c

= SETVAR command: struct for args was too short: using GLOBALONLY saved the
  variable to ENVARC:, using SAVE caused a crash. removed LOCALONLY/S and
  replaced GLOBALONLY/S by GLOBAL=GLOBALONLY/S (as described in the docs)
  SAVE now only works on AmigaOS 3.0 and above
  IPDial.c/SetVarFunc(), removed setvar.c

+ compiled with NOCHECKABORT (SAS/C), so Cleanup() is always called when
  IPDial terminates

+ DELAY now takes two arguments (seconds and milliseconds) and can be
  aborted with Ctrl-C (exit) and Ctrl-F (terminal mode)
  IPDial.c/DelayFunc(), added Serial.c/TimerWait()

+ When IPDial is waiting for a string from serial.device and aborted with
```

```
  Ctrl-C, CRLF is sent to hangup the modem in case it was dialing
  Serial.c/SerialWait()

- no more Delay() after DoIO (as in IPDial 1.8/1.9). This delay made the
  character terminal mode unusable. Compile with DELAY_AFTER_IO defined
  (IPDial.h) to use a delay again.
  DeviceIO.c/DeviceIODo()

= ECHO, SEND and SYSTEM now use ARGS/F instead of ARGS/M -> arguments are
  one single string. Adapted ParseString() to remove extra quotes.
  IPDial.c/EchoFunc(), IPDial.c/SendFunc(), IPDial.c/SystemFunc(),
  IPDial.c/ParseString()

- no more parameter "Length" in SerialSend (it wasn't used anyway)
  Serial.c/SerialSend()

+ command line option ECHO now works for terminal mode, too
  IPDial.c/main()

- removed HELP/S command line option ("?" will do, too)

+ VERBOSE is now a bit more verbose
  Serial.c/SerialOpen(), IPDial.c/WaitFunc(), IPDial.c/*

- removed Buffer.c/BufferExpand() - was never called

+ $VAR and ${VAR} now work for local (shell) and global variables
  IPDial.c/ParseString()

+ pressing Ctrl-F while a DELAY or WAIT command is running gets you into
  terminal mode (character mode without local echo)
  Serial.c/SerialWait(), Serial.c/TimerWait()

+ Ctrl-C now exits terminal mode with RC 5, Ctrl-F with RC 0 (useful
  for shell scripts)
  Serial.c/SerialTerminal()

= added global scratch buffer -> many local buffers removed
  IPDial.c/ParseFile(), IPDial.c/ParseString()

+ included second example dial script, improved hangup script

= rewrote documentation (now AmigaGuide)


Changes in IPDial 2.1 (16.04.96)

= fixed a couple of errors in the documentation (please reread the Updating
  and Common problems sections)

+ included dial script for the German T-Online (examples/T-Online.IPDial)

= EXIT now checks its arguments
  IPDial.c/ExitFunc()

+ the serial buffer size is now set to 4096 bytes when a device is opened.
  Serial.c/SerialOpen()
```

```
+ cleaned up code a little bit (executable is now 2 KB smaller)
  IPDial.c/PrintError(), IPDial.c/CleanupAndExit()

- removed SET and SHOWPARAMS commands. Some options from the SET command were
  moved to the DEVICE command.
  IPDial.c/SetFunc(), IPDial.c/ShowParmsFunc(), IPDial.c/DeviceFunc(),
  Serial.c/SerialOpen(), Serial.c/SerialShowParms(),
  Serial.c/SerialSetDataBits(), Serial.c/SerialSetStopBits(),
  Serial.c/SerialSetParity(), Serial.c/SerialSetProtocol()

= TIMEOUT in WAIT command is now optional (default: 5 seconds)
  IPDial.c/WaitFunc()

+: new feature
=: bugfix / change
-: removed feature
```

1.22 Some ideas for the future of IPDial

```
The next steps in IPDial will be a logfile (suggestions welcome) and support
for OwnDevUnit.library. If you have any other enhancement requests, please
send me an email.
```

1.23 Thanks

```
Michael 'Mick' Hohmann, Angela Schmidt

    for their icons (taken from the Meeting Pearls III CD ROM)


Kai Kohlmorgen

    for the T-Online dial script


and everybody else who sent me bugreports and suggestions.
```

1.24 How to reach the authors

```
IPDial 1.1 to 1.7 was written by

    Jochen Wiedmann <wiedmann@neckar-alb.de>
    WWW: http://www.neckar-alb.de/iss/jochen/


Changes in IPDial 2.x by

    Stefan Gybas <cab@studbox.uni-stuttgart.de>
    WWW: http://home.pages.de/~cab/
```

IPDial WWW support pages

http://wwwcip.rus.uni-stuttgart.de/~inf11108/support/ipdial.html