Miami

COLLABORATORS

	TITLE : Miami	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Miami

1.1 Miami.guide

```
                Miami
*****

   This is the documentation for Miami V1.1, an integrated TCP/IP
system for AmigaDOS. Copyright (C) 1996 Holger Kruse.  All rights
reserved. Documentation by Holger Kruse.
```

1.2 Miami.guide/NODE_DISCLAIMER

Disclaimer
****

Miami IS SUPPOSED TO BE A TCP/IP PACKAGE FOR AmigaDOS THAT CAN BE USED TO CONNECT YOUR AMIGA TO THE INTERNET BY MODEM. EVEN THOUGH EVERY EFFORT HAS BEEN MADE TO MAKE Miami AS COMPATIBLE TO THE TCP/IP STANDARD AS POSSIBLE, I CANNOT RULE OUT THE POSSIBILITY THAT Miami HAS BUGS THAT HAVE HARMFUL SIDE EFFECTS ON YOUR SYSTEM OR ON OTHER MACHINES CONNECTED TO YOUR AMIGA.

I HEREBY REJECT ANY LIABILITY OR RESPONSIBILITY FOR THESE OR ANY OTHER CONSEQUENCES FROM THE USE OF Miami WHATSOEVER. THIS INCLUDES, BUT IS NOT LIMITED TO, DAMAGE TO YOUR EQUIPMENT, TO YOUR DATA, TO OTHER MACHINES YOUR AMIGA IS CONNECTED TO, ANY EQUIPMENT CONNECTED TO THAT HOST, PERSONAL INJURIES, FINANCIAL LOSS OR ANY OTHER KINDS OF SIDE EFFECTS.

Miami IS PROVIDED AS-IS. THIS MEANS I DO NOT GUARANTEE THAT Miami IS FIT FOR ANY SPECIFIC PURPOSE AND I DO NOT GUARANTEE ANY BUG FIXES, UPDATES OR HELP DURING ERROR RECOVERY.

Miami is based on the 4.4BSD V.2 TCP/IP networking code, in the version distributed by Walnut Creek on CD-ROM.

All of the original 4.4BSD code is freely distributable, and has been contributed by different sources. For details about individual copyright and disclaimer rules, please refer to the source files, which are available from different sources, e.g. from the 4.4BSD Lite CD-ROM available from Walnut Creek.

The following copyright notice applies to the complete original 4.4BSD software package:

Start quote

All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by The Regents of the University of California.

Copyright 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994 The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors. 4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

End Quote

Please be advised that this copyright notice does NOT apply to the Miami package. Miami is NOT freely distributable, unless otherwise stated. See
Usage / Copying
for details.

Miami relies on Magic User Interface (MUI). MUI is Copyright by Stefan Stuntz.

Miami requires the MUI custom class "Busy.mcc' by Klaus Melchior.

Here is the associated copyright notice:

   Begin Quote

   Busy.mcc is (c) 1994-1996 by Klaus 'kmel' Melchior

   End Quote

1.3 Miami.guide/NODE_CONDITIONS

Usage / Copying
****

   Miami is shareware. In this case this means that a personalized key file is required to use the full functionality of Miami.

   Users will receive their personalized key file from me after registering. The key file may not be made available to other users ! Giving the key file to other users or using key files that you did not receive directly from me for your personal use is considered an act of software piracy !

   The Miami binary or the binaries of any of the utility programs may not be modified or patched in any way (not even for personal use), except in ways explicitly approved by me for software updates. Using patched or modified binaries is considered an act of software piracy !

   Miami binaries may only be used for the purpose intended, i.e. to be executed on Amiga computers by AmigaOS. Reassembling, reverse-engineering, or translating binaries is expressly prohibited.

   The documentation and program texts of Miami are subject to the same copyright as the program itself. This means neither documentation nor program texts may be modified or translated in any way.

   To avoid any misunderstanding: YOU MAY NOT translate and distribute Miami program texts or documentation, unless I officially appoint you as a Miami translator. Unauthorized translations of program texts or documentation are illegal, violate my copyright, and will be deleted from public software sites.

   If you want to distribute the Miami archive the following conditions apply:

   * The sales price must not be higher than the cost of the empty disks required for the Miami files plus a nominal copying fee plus costs for shipping. The total price must not be higher than 10 US$ or 15 DM or the equivalent in any other currency.

   * If the Miami archive is to be distributed as part of a CD-ROM collection of public domain and/or shareware programs, then the retail price of the CD-ROM may not exceed 20 US$, 30 DM or the equivalent in any other currency.

* All parts of the program and the documentation must be complete. The distribution of single parts or incomplete subsets of the original distribution is not allowed. The distribution of keyfiles is not allowed.

* Miami or parts of it may usually not be sold in combination with or as part of commercial software. Separate licensing conditions for commercial resale are available from kruse@nordicglobal.com upon request. However, unless and until you receive my explicit written approval, do not assume that you may distribute Miami or parts of it in combination or as part of commercial software.

* Program and documentation may not be changed in any way. Exception (this means: acceptable) is the use of archivers such as LHA as long as it remains possible to retrieve the original program/data.

1.4 Miami.guide/NODE_REGISTRATION

Registration
****

If you often use Miami, need any of the features disabled in the demo version, or want to stay connected for more than one hour at a time, I suggest you register Miami.

To register please run the program MiamiRegister. It explains the registration procedure in detail, and allows you to register interactively.

If you for some reason cannot run MiamiRegister please contact me at kruse@nordicglobal.com.

The registration fee is US$ 35 for a standard, 'full' Miami license. It is also possible to obtain a 'limited' Miami license that only works with a single Internet provider, if this provider is participating in the Miami promotion program.

Registered users of ppp.device receive a discount when upgrading to Miami. The details are explained by MiamiRegister.

Special offers for group licensing (10 users or more at a time), license prepayment and commercial redistribution are also available. Please contact kruse@nordicglobal.com for more details.

1.5 Miami.guide/NODE_INTRODUCTION

Introduction
****

Miami is an integrated TCP/IP system for AmigaDOS, that allows you to access the Internet by modem in a very simple way.

Miami is based on the latest version (4.4BSD V2) of the official BSD networking code, i.e. Miami contains a "true" and complete TCP/IP stack, not just an emulation that only supports parts of the TCP/IP standard.

The application programmers' interface of Miami is compatible with that of AmiTCP 4.x (bsdsocket.library), i.e. most of the programs written and compiled for AmiTCP 4.x will work with Miami without any modification and without recompiling.

In addition, Miami has a built-in dialer that can be used both in script-driven and interactive mode, an implementation of the (C)SLIP and PPP protocols and a graphical user interface for program control and configuration.

Miami also has a built-in implementation of inetd, the "Internet super-server", with several built-in services including "fingerd" and "identd", a built-in implementation of TCP:, the AmigaDOS stream handler for TCP/IP, and a built-in implementation of usergroup.library, the interface to manage users and user groups.

Unlike other general-purpose protocol stacks Miami is specifically targetted towards users who use a modem to access the Internet. The configuration process is made as simple as possible: most of the configuration parameters are determined automatically by Miami. Miami also supports preconfigured settings that can be distributed by Internet providers.

Miami uses MUI 3.3 or higher for its user interface, i.e. you must have MUI installed before you can use Miami.

You should have a look at
MiamiInit
. MiamiInit is a program that for
most users automatically configures Miami to your needs, including dial script, authentication, IP address, DNS servers and all other configuration variables.

After running MiamiInit you should run Miami, import the configuration from MiamiInit, save the new settings, and connect to your provider.

1.6 Miami.guide/NODE_REQUIREMENTS

Requirements
****

To use Miami you need:

* an Amiga running OS 2.04 or higher

* MUI 3.3 or higher

* a modem connected to your Amiga and to a phone line. The modem should be at least roughly Hayes-compatible. Most contemporary modems are.

* a SLIP or PPP account with an Internet provider. If you only have a shell account you can use Miami as well, but then you need to install Slirp or TIA at your provider first. In this case you should ask your provider whether you are allowed to do this, and how and where you can get Slirp or TIA.

1.7 Miami.guide/NODE_INSTALLATION

Installation
****

No installation is required. Miami does not require any assigns, environment variables etc.

If you received Miami by downloading an archive such as "Miami10.lha" then simply extract all files from the archive to the directory where you want Miami to be installed.

If you received Miami on a CD-ROM you might want to copy all files to your harddisk first. To do this type
copy <sourcedir> <destdir> all
in a Shell window.

1.8 Miami.guide/NODE_MIAMIINIT

MiamiInit
****

MiamiInit is a utility program that tries to determine all configuration parameters for Miami, and then saves a configuration file that can later be used by Miami.

The first thing you should do after installing the Miami package is to run MiamiInit, and go through the dialog. In the process MiamiInit dials up your Internet provider, determines all required parameters, and saves them at the end.

MiamiInit only supports the most common setups at the moment. Very unusual cases such as data formats other than 8N1, non-Hayes-compliant modems or 3-wire modem cables are not supported. If you have any such unusual setup you need to configure Miami manually instead of running MiamiInit.

1.9 Miami.guide/NODE_TOOLTYPES

```
ToolTypes
*********

   Miami supports the following ToolTypes when started from Workbench
(or arguments when started from the Shell):

DONTCONNECT
     If you have configured Miami to automatically connect to your
     Internet provider whenever you start Miami, then you can use this
     ToolType to override that behavior, giving you a chance to change
     some settings before you connect.

SETTINGS
     Any project icon needs to have a "SETTINGS" ToolType so Miami
     recognizes it as a settings file. From the Shell you can use the
     argument "SETTINGS=filename" to specify the settings file to load.

IMPORTMIAMIINIT
     The argument "IMPORTMIAMIINIT=filename" tells Miami to import a
     settings file from MiamiInit.

IMPORTASCII
     The argument "IMPORTASCII=filename" tells Miami to import an ASCII
     settings file (distribution format).

SAVESETTINGS
     The argument "SAVESETTINGS" tells Miami to save the settings as
     default.  This argument is most useful when combined with
     "IMPORTMIAMIINIT" or "IMPORTASCII" to import a foreign settings
     file and convert it to a Miami settings file.
```

1.10 Miami.guide/NODE_CONFIGURATION

```
                Configuration
*************

   The configuration of Miami is done completely through the graphical
user interface. There are no configuration files or environment
variables to edit.

   Description of the graphical user interface:



                 General
                                    The 'General' page
```

Interface
The 'Interface' page

PPP
The 'PPP' page

Dialer
The 'Dialer' page

Database
The 'Database' page

TCP/IP
The 'TCP/IP' page

Events
The 'Events' page

Modem
The 'Modem' page

Logging
The 'Logging' page

GUI
The 'GUI' page

Misc
Other GUI elements

1.11 Miami.guide/NODE_GUI_GENERAL

General
=======

Not much here, except for the official Miami logo and a gadget to start the Miami registration program.

Register
The 'Register' gadget

1.12 Miami.guide/NODE_GUI_GENERAL_REGISTER

Register

This gadget starts the program MiamiRegister, allowing you to order a Miami license code, register Miami or upgrade your registration. MiamiRegister has to be in the same directory as Miami, or in the standard Shell path.

1.13 Miami.guide/NODE_GUI_INTERFACE

Interface
=========

Device / Unit
The 'Device' and 'Unit' gadgets

Speed
The 'Speed' gadget

Protocol
The 'Protocol' gadget

Flow control
The 'Flow control' gadget

EOF mode
The 'EOF mode' gadget

Serial mode
The 'Serial mode' gadget

MTU
The 'MTU' gadget

IP type / address
The 'IP' gadgets

Use CD
The 'CD' gadget

Use BootP
The 'BootP' gadget

Inactivity
The 'Inactivity' gadgets

1.14 Miami.guide/NODE_GUI_INTERFACE_DEVICE

Device / Unit

Enter the device name and unit number of the serial port to which your modem is connected. For the built-in Amiga serial port use 'serial.device' '0'.

For serial boards use the driver that comes with the board, e.g. 'gvpser.device', with the correct unit number.

Other device drivers for the internal serial port like 'v34serial.device' are supported, too. You should not use '8n1.device' at this time though, because of bugs in the device. Some users have also reported problems with 'BaudBandit.device'.

1.15 Miami.guide/NODE_GUI_INTERFACE_SPEED

Speed

Speed of your serial port. For the internal serial port you should use 19200, 38400 or (if you have a fast CPU and a graphics board) 57600. For serial boards you might even be able to use 115200 or 230400.

Do not use 31250. This speed is reserved for MIDI only and usually does not work with modems.

Do not use 14400, 28800 or 33600 either. Your modem might be able to connect to the other modem at these speeds, but it does probably not support these speeds on its serial port.

1.16 Miami.guide/NODE_GUI_INTERFACE_PROTOCOL

Protocol

The protocol your Internet provider uses. Currently supported are SLIP/CSLIP and PPP.

1.17 Miami.guide/NODE_GUI_INTERFACE_FLOW

Flow control

Miami supports two types of flow control: hardware handshaking (RTS/CTS) and software handshaking (Xon/Xoff). By default hardware handshaking is used, and it is strongly recommended that you do not change this.

If you cannot use hardware handshaking (usually because of a defective modem, cable or serial port) you should switch to software handshaking. However make sure that you change your modem init string (in the dialer window) appropriately. Also, software handshaking is only possible with PPP, not with SLIP/CSLIP.

1.18 Miami.guide/NODE_GUI_INTERFACE_EOF

EOF mode

There are two ways for Miami to detect the end of incoming packets: The more efficient one (using less CPU time) uses the EOF_MODE flag. However this is only possible if the serial driver you use supports EOF mode. Many third-party drivers do not.

Usually you should leave this switch in the "auto" setting to let Miami use the default setting. If you positively know whether your driver supports EOF-mode or not you can manually override the default setting by choosing "on" or "off".

1.19 Miami.guide/NODE_GUI_INTERFACE_SERIAL

Serial mode

The settings for the number of data bits and parity used during dialing. For 99% of all providers the correct settings are 8N1. Very few providers (e.g. some dialin points for Compuserve) might require 7E1 or 7O1.

Please note that these settings only apply during dialing and login. The (C)SLIP/PPP protocol phases always use 8N1, regardless of the setting you specified here. It is completely impossible to use PPP or (C)SLIP across a 7-bit line - with any implementation actually. This is not a limitation in Miami.

1.20 Miami.guide/NODE_GUI_INTERFACE_MTU

MTU

Maximum Transfer Unit, i.e. the size of the largest packet transferred at a time. The default for PPP is 1500, the default for (C)SLIP is 1006.

For serial lines it is usually a good idea to use a smaller MTU value

than the default in order to get a better response time for interactive TCP/IP traffic. Good values are:

* for modem speeds up to 19200 bps: MTU=296

* for modem speeds higher than 19200 bps: MTU=552

Please note that changing the MTU value in the configuration window does not necessarily mean that the maximum packet size is actually changed to this value:

(C)SLIP does not have any means to negotiate MTU, i.e. the MTU value configured here only affects the size of outgoing packets, not the size of incoming packets.

PPP has configuration options to negotiate the MTU. Miami always tries to negotiate the MTU you specified here, but the other side might disagree and force a different MTU value, in which case Miami might have to use the value suggested by the other side for one or both directions.

1.21 Miami.guide/NODE_GUI_INTERFACE_IP

IP type / address

Internet providers usually offer two types of Internet connections: those with a static IP address permanently assigned to your Amiga, or (more popular) those where your Amiga receives a dynamic IP address each time you connect.

In the first case choose "static" and enter the IP address your provider told you. In the second case choose "dynamic", and Miami determines the IP address automatically when you connect.

If you use TIA or Slirp you have to choose "static" and enter the pseudo IP address that TIA or Slirp assign to your Amiga. Please see the TIA/Slirp docs for more information about this.

1.22 Miami.guide/NODE_GUI_INTERFACE_CD

Use CD

If "Use CD" is activated then Miami uses the "Carrier Detect" line of your modem to determine if your modem is already connected to the other side or not.

This can be useful if you reset your Amiga without dropping the line, so you can restart Miami and reconnect to your provider without

redialing.

This option can only be used if your modem has been configured to correctly set the "Carrier Detect" line according to the line state.

Some modems have factory default settings that always set the "Carrier Detect" line to high, even if the modem is not connected. If this is true for your modem then you either have to change the modem settings in your modem init string (usually "AT&C1"), or switch off the "Use CD" option.

If you are using the null-modem settings (configured on the "Modem" page) then this gadget gets a different meaning:

* If the gadget is activated then the dial script is not executed at all.

* If the gadget is deactivated then the dial script is executed, except that Miami does dial a number, i.e. the "ATDT..." command is skipped, and the list of phone numbers is meaningless.

1.23 Miami.guide/NODE_GUI_INTERFACE_BOOTP

Use BootP

If your provider uses dynamic IP addresses then there are different techniques for Miami to find the correct (dynamic) IP address.

For PPP lines this is usually handled has part of the PPP protocol. (C)SLIP does not have such an option though, so for (C)SLIP a protocol called "BootP" is sometimes used. Alternatively the IP address can sometimes be determined from the dial log.

If you used MiamiInit to configure the line then you can just leave this switch at its default setting. If you configured Miami manually then you should first switch "BootP" on, and then later try again with "BootP" switched off, and see if this still works.

If Miami can find your IP addresses without BootP then you should switch "BootP" off, because it can make the connection establishment phase quicker.

1.24 Miami.guide/NODE_GUI_INTERFACE_INACTIVITY

Inactivity

Some Internet providers hang up the line if there is no activity on the line for a while to prevent users from occupying lines that are not

really used.

The "Inactivity" gadgets allow you to configure Miami to simulate line activity even if you are not really using the line, so your provider does not hang up.

The gadget on the left sets the type of activity: PPP ping or ICMP ping. PPP ping consumes less bandwidth, but only works with PPP, not with (C)SLIP, and does not have an effect with all providers. ICMP ping takes up slightly more bandwidth, but works with both PPP and (C)SLIP, and should have an effect with all providers.

If you use (C)SLIP then choose ICMP ping. Otherwise first try PPP ping, and if your provider still hangs up try ICMP ping.

The gadget on the right sets the number of minutes between successive pings. You need to experiment with that. Common values are 9 or 14, to prevent hangups after 10 or 15 minutes.

Note: You need to check with your Internet provider first if he allows the use of this type of activity simulator. Some providers have policies that do not allow it, and by using such a simulator you might be violating their regulations. I will not be responsible or liable for any consequences resulting from the improper use of this activity simulator.

Note: There are many reasons why a modem might hang up. One is an inactivity timeout at your Internet provider, which should be prevented by this function. However modems sometimes also hang up the line because of line noise. There is no way to prevent this in software.

1.25 Miami.guide/NODE_GUI_PPP

PPP
===

PAP / CHAP password
The 'PAP/CHAP' gadgets

VJC
The 'VJC' gadget

ACCM
The 'ACCM' gadget

Quick Reconnect
The 'Quick Reconnect' gadget

Get DNS from IPCP
The 'Get DNS from IPCP' gadget

Escape

The ‘Escape’ gadget

1.26 Miami.guide/NODE_GUI_PPP_CHAP

```
PAP / CHAP password
-------------------
```

PAP and CHAP are protocols used by PPP to send login id and password to the PPP server.

Most of the time the login id and password used for PAP or CHAP are identical to the ones you used in your dial script. In this case choose "Same as in dialer".

If your provider requires a PAP/CHAP login id or password different from the one you chose in the dialer, then do not select "Same as in dialer", but instead type in your PAP/CHAP login id and password manually.

1.27 Miami.guide/NODE_GUI_PPP_VJC

```
VJC
---
```

Van Jacobsen Compression is a technique to save bandwidth by compressing the headers of TCP packets. Usually this option should be switched on, meaning that PPP will automatically try to negotiate VJC, and use it if the other side agrees.

However some old, buggy PPP servers do not support VJC properly, so you might have to switch VJC off for them.

VJC does not interact with your modem’s data compression in any way, i.e. you should not switch VJC off just because your modem supports MNP-5 or V.42bis. VJC can be used independently of MNP-5 or V.42bis.

1.28 Miami.guide/NODE_GUI_PPP_ACCM

```
ACCM
----
```

The PPP protocol supports a list of control characters that are "escaped" during transmission, i.e. replaced by a two-byte sequence. This list is called ACCM (Asynchronous Control Character Mask).

The purpose of this list is to make PPP more robust across lines

that are not completely 8-bit transparent, and to avoid any interference of the PPP protocol with software modem flow control.

The default is to only escape characters 17 and 19 (Xon/Xoff), so PPP can be used across a link with software flow control. If you are running PPP through a telnet link you might have to escape more characters. Each character you escape reduces the performance of PPP by about 0.8%.

To change the ACCM settings either enter the 32-bit mask value directly in heaxdecimal digits, or click on the popup gadgets to toggle each control character individually.

1.29 Miami.guide/NODE_GUI_PPP_QUICK

Quick Reconnect

Usually Miami allows you to reconnect to your provider (without dialing again) when the modem is still connected, e.g. after resetting your Amiga, but only if the "Use CD" gadget is switched on on the "Interface" page.

However even then with PPP some providers do not allow reconnection (and renegotiation of PPP), and instead hang up the line when you try to reconnect.

"Quick Reconnect" usually helps in this case: If "Quick Reconnect" is activated then Miami does not attempt to renegotiate PPP, but bypasses the renegotiation and fetches all PPP parameters from an area of RAM that has been setup to survive a reboot. In most cases this allows you to reconnect to your provider after rebooting your Amiga.

However this technique only works if you do not reboot at all, or after a soft- (warm-) reboot. If your machine crashes very badly or if you have to cold-reboot (destroying resident modules) then the old PPP parameters will be gone and "Quick Reconnect" does not work.

1.30 Miami.guide/NODE_GUI_PPP_DNSIPCP

Get DNS from IPCP

This switch is "on" by default. This means that Miami tries to use IPCP extensions for automatic DNS discovery to find DNS servers.

Unfortunately some broken PPP servers neither support this option, nor reject it properly, but simply violate the protocol. If you experience problems completing the link level PPP protocol with your Internet provider you might have to disable this option.

1.31 Miami.guide/NODE_GUI_PPP_ESCAPE

```
Escape
------
```

PPP can negotiate that characters in the range of 0-31 and 128-159 are escaped. This is configured in the ACCM.

However there are situations when you might have to escape some additional characters, e.g. 0xFF across rlogin connections.

In this case enter the 2-digit hex codes (separated by spaces) into the "Escape" gadget, and Miami will escape those characters when sending PPP packets.

Note that, contrary to the ACCM definition, this only works in one direction: when sending data. If the channel back from the server to Miami also requires character escaping, then you have to configure the PPP server accordingly as well.

1.32 Miami.guide/NODE_GUI_DIALER

```
                Dialer
======
```

Dial script
: The ‘Dial script’ listview

Phone numbers
: The ‘Phone numbers’ listview

Max Repeat
: The ‘Max Repeat’ gadget

Repeat Delay
: The ‘Repeat Delay’ gadget

Teach
: The ‘Teach’ gadget

Login ID / Password
: The ‘Login ID’ / ‘Password’ gadgets

Capture
: The ‘Capture’ gadgets

1.33 Miami.guide/NODE_GUI_DIALER_SCRIPT

```
                Dial script
-----------
```

   There listview gadget in the top area of the "Dial script" group contains the dial script. You can change entries by clicking on them and editing them in the string gadget below.

   The gadgets at the bottom are used to add and remove entries from the dial script.

   For more information about the language used by the dialer please see

                Dialer Command Language
                .

   The listview has a context menu associated with it, i.e. by pressing the right mouse button over the listview a menu pops up allowing you to import/export the dial script from/to an ASCII text file.

1.34 Miami.guide/NODE_GUI_DIALER_PHONE

```
Phone numbers
-------------
```

   The "Phone numbers" group works similarly to the "Dial script" group, but has two additional gadgets: "Enable" and "Disable". Enabled phone numbers have a "»" symbol next to them. Only enabled phone numbers will be used during dialing.

   In the demo version only up to three phone numbers are supported. In the registered version there is no such limit.

1.35 Miami.guide/NODE_GUI_DIALER_MAX

```
                Max Repeat
----------
```

   If no connection can be established with any of the listed phone number, then Miami waits for the time specified in
                Repeat Delay
                , and
then tries again, restarting with the first phone number. However the maximum number of retries is limited by the number specified in the "Max Repeat" gadget. After that Miami just gives up and aborts dialing.

1.36 Miami.guide/NODE_GUI_DIALER_DELAY

```
Repeat Delay
------------
```

If no connection can be established with any of the listed phone number, then Miami waits for the time specified in the "Repeat Delay" gadget and then tries again, restarting with the first phone number.

1.37 Miami.guide/NODE_GUI_DIALER_TEACH

```
Teach
-----
```

The "Teach" gadget starts the Miami dialer in interactive mode (i.e. without executing a dial script), records all text send by the user or received from the modem, and then tries to create a proper dial script from that.

Most of the time MiamiInit is used to create a dial script, not "Teach", but if your provider changes the login procedure it might be more convenient for you to only create a new dial script (using "Teach") instead of running MiamiInit all over again.

1.38 Miami.guide/NODE_GUI_DIALER_NAME

```
Login ID / Password
-------------------
```

The login id and password used in the dial script. If "Same as in dialer" is enabled in the PPP window then these values are also used for PAP/CHAP.

1.39 Miami.guide/NODE_GUI_DIALER_CAPTURE

```
Capture
-------
```

If you activate the "Capture" checkmark gadget and enter a file name in the corresponding string gadget, then the dialer will save all data received from the modem during dialing (i.e. a complete dial log) to a file.

1.40 Miami.guide/NODE_GUI_DATABASE

```
                Database
========
```

The "Database" page is the equivalent of the files in the "db" directory for other TCP/IP protocol stacks, i.e. it allows you to configure most of the TCP settings on your system, which daemons to run, a list of users and other things.

The cycle gadget on top of the listview is used to switch between different parts of the database. There is a template below the listview describing how proper entries have to look like. A more detailed explanation follows below.

Using the context menu of the database listview gadget you can import/export each part of the database from/to ASCII text files. This allows you to continue to use your old AmiTCP/AS-225 db/#? files with Miami.

Each entry in the database can be marked as "temporary" by clicking on the "Temp" gadget. This has the effect that this entry is not saved to disk when you save the settings, and that it is - in some cases - deleted when reconnecting. This can be useful if some of the entries (e.g. dynamically obtained DNS server addresses) should not be used for the next connection.

By default Miami marks all dynamically obtained DNS server addresses and your dynamic hostname as temporary.

The "Change Password" gadget can only be used together with entries in the "Users" part of the database. It allows you to change the password for a user and automatically encrypts it.

Parts of the database:

```
                Protocols
                                        The ‘protocols’ part

                Services
                                         The ‘services’ part

                Hosts
                                            The ‘hosts’ part

                Networks
                                         The ‘networks’ part

                Domains
                                          The ‘domains’ part

                DNS servers
                                       The ‘DNS servers’ part
```

InetD
The 'InetD' part

users
The 'users' part

groups
The 'groups' part

1.41 Miami.guide/NODE_GUI_DATABASE_PROTOCOLS

Protocols

List of all supported protocols (relative to IP) in the format

protocol_name protocol_id optional_list_of_aliases

This table hardly ever has to be changed.

1.42 Miami.guide/NODE_GUI_DATABASE_SERVICES

Services

List of all supported services (TCP or UDP) in the format

service_name service_id/protocol_name optional_list_of_aliases

Some application programs might require changes to this list.

1.43 Miami.guide/NODE_GUI_DATABASE_HOSTS

Hosts

List of all host names (and corresponding IP addresses) in the format

ip_address host_name optional_list_of_aliases

Miami automatically adds a mapping for "localhost" and for the host name of your Amiga to this list. Other mappings can be added manually to make name->IP translations faster.

1.44 Miami.guide/NODE_GUI_DATABASE_NETWORKS

```
Networks
--------

   List of all networks in the format

     network_name network_id optional_list_of_aliases

   This table is hardly used any more, and only implemented for
backwards compatibility with very old software.
```

1.45 Miami.guide/NODE_GUI_DATABASE_DOMAINS

```
Domains
-------

   List of all local domains in the format

     domain_name

   This table is not strictly needed by TCP/IP, but adds some
convenience for the user: it allows you to abbreviate host names by
specifying just the machine name (without the domain) whenever
referring to a host.

   Example:

   Assume a local machine on your network is named ex1.foo.edu, and you
access this machine frequently. If you add foo.edu to the list of
domains, then you can access machine ex1.foo.edu by just typing ex1.
```

1.46 Miami.guide/NODE_GUI_DATABASE_DNSSERVERS

```
DNS servers
-----------

   List of DNS servers in the format

     IP_address

   DNS servers are used to map logical host names to their IP address.
You should have at least one DNS server listed in this table at all
times, preferably a DNS server close to or at your provider.

   If Miami finds any DNS servers by itself when connecting it
automatically adds them to this list and marks them as "temporary".
```

1.47 Miami.guide/NODE_GUI_DATABASE_INETD

```
InetD
-----

   List of daemons started by the built-in InetD in the format

     service_name socket_type protocol_name wait_mode owner file_name args

   "socket_type" is one of "dgram" or "stream". "wait_mode" is one of
"wait", "nowait" or "dos".

   The InetD built-in to Miami supports man built-in services:
"daytime", "time", "echo", "discard", "chargen", "finger" and "auth".
"auth" is really the same as "identd".

   Daemons for other (external) services can be automatically started
by InetD by adding an appropriate line to this table. If you would like
to install external daemons (e.g. ftpd or telnetd) please check their
documentation for the correct format of the "InetD" entry they require.
```

1.48 Miami.guide/NODE_GUI_DATABASE_USERS

```
                Users
-----

   List of users in the system in the format

     user_name|password|user_id|group_id|real_name|home_dir|shell

   You usually only need a single entry in this file (for yourself),
unless you want to run daemons like ftpd/telnetd that allow other users
to connect to your Amiga.

   Passwords are stored in an encrypted format and should not be
changed manually.  To change a password, first select a user entry in
the listview and then click on "Change password". This will prompt you
to enter the new password, and store it in encrypted format.

   Note: When you import this file from AmiTCP the passwords are not
preserved, i.e. the passwords for all users are set to empty and have
to be entered again manually. This is because the password encryption
algorithm used by AmiTCP cannot be used by Miami for legal reasons. For
more information on this please check
                Passwords
                .
```

1.49 Miami.guide/NODE_GUI_DATABASE_GROUPS

Groups

   List of groups in the system in the format

     group_name|*|group_id|optional_user_list

   You usually only need a single entry in this file (for yourself),
unless you want to run daemons like ftpd/telnetd that allow other users
to connect to your Amiga.

   The "*" represents an unused password entry. Passwords are not
currently supported for groups.

1.50 Miami.guide/NODE_GUI_TCPIP

                TCP/IP
======

Host name
    The 'Host name' group

Real / User name
    The 'Real name' and 'User name' gadgets

Use ICMP
    The 'Use ICMP' gadget

Fake IP
    The 'Fake IP' gadget

Verify DNS servers
    The 'Verify DNS servers' gadget

Auto-add domain
    The 'Auto-add domain' gadget

Get time
    The 'Get time' gadgets

1.51 Miami.guide/NODE_GUI_TCPIP_HOSTNAME

Host name

In most cases you should switch the gadget "dynamic" on. In this case Miami automatically determines your Amiga's host name through reverse DNS lookup whenever you connect.

However some providers do not support reverse DNS lookup, or assign a static host name to their user that is not listed in the DNS. In this case switch "dynamic" off and enter your host name manually.

1.52 Miami.guide/NODE_GUI_TCPIP_NAME

Real / User name

In these gadgets you should enter your real name (e.g. "Joe Smith"), and the user name on your Amiga (e.g. "jsmith").

Although you could theoretically use any names here it is good practice to use "real" names, not some phantasy names.

Some programs look up user information based on your user name. To make these programs behave properly you should ensure that there is an entry in the "Users" part on the "Database" page that corresponds to the user name entered here.

1.53 Miami.guide/NODE_GUI_TCPIP_ICMP

Use ICMP

If this gadget is switched on then Miami uses ICMP "ping" messages to verify the correctness of IP addresses, DNS servers etc.

This gadget should usually be switched on, because it provides additional protection from incorrect configuration.

However if you are connecting through some TCP emulator such as TIA then you might have to switch this gadget off, because not all TCP emulators support ICMP.

1.54 Miami.guide/NODE_GUI_TCPIP_FAKEIP

Fake IP

If you are connected to the Internet through a TCP emulator such as TIA or Slirp, and this emulator does not assign you a "real" IP address, but a fake address, then you need to activate this switch.

It tells Miami to obtain your host name by resolving the remote IP address, not your local ("fake") IP address.

1.55 Miami.guide/NODE_GUI_TCPIP_VERIFYDNS

```
Verify DNS servers
------------------
```

Usually Miami tries to verify the correctness of the IP addresses of all DNS servers. However this can cause problems with some Internet providers if their DNS servers have a bad connectivity of do not respond to requests immediately after connection establishment.

If you deactivate the "Verify DNS servers" gadget then Miami skips the DNS verification step when going online.

1.56 Miami.guide/NODE_GUI_TCPIP_ADDDOMAIN

```
                Auto-add domain
---------------
```

If this gadget is activated then Miami will automatically add your host name's domain (i.e. everything after the first '.') to Miami's "domain" database.

This is not strictly required for Miami or for any software, but it can be convenient if you want to use abbreviated host names. Please see

```
                The 'Database' page
                 for more details on the meaning of the "domain"
database.
```

1.57 Miami.guide/NODE_GUI_TCPIP_GETTIME

```
Get time
--------
```

If your Amiga is not equipped with a battery-powered real-time clock then you should activate the "Get time" switch, and enter the name or IP address of a server that supports the "time" service in the string gadget. If you are unsure which name to enter just try any "major" machine run by your provider, e.g. the machine you use for e-mail or news.

If you use this feature you need to make sure that your "ENV:TZ"

variable is set correctly, i.e. usually to something like "EST5", or to
"EST4EDT" during daylight savings time. This is important, because the
server transmits the time in GMT (UTC) format, and Miami needs to
adjust it to your local time zone.

1.58 Miami.guide/NODE_GUI_EVENTS

Events
======

Miami allows you to react in various ways to events such as offline, online etc., by executing an ARexx script, iconifying the Miami window etc.

The specific events Miami can react to are:

Start
: program start.

End
: program end.

active Offline
: going offline caused by the user, e.g. by clicking on the "Offline" gadget or by an ARexx "OFFLINE" command.

passive Offline
: going offline caused by the modem or the provider hanging up.

Online
: going online, i.e. successfully connecting to the Internet provider and starting up all required protocols.

failed Online attempt
: an attempt to go online that failed for some reason, e.g. because all phone lines were busy, and the maximum number of retries was reached.

Miami can react in the following ways. Not each of these options makes sense for each event, so only a subset of these options is actually available in each case:

ARexx
: Start an ARexx script

hide
: iconify the Miami window

auto-online
: try to go online (dial) automatically

beep
: flash the display or beep, as defined in system preferences

show
     deiconify the Miami window

   In the evaluation version of Miami the option "ARexx" is not available, and "auto-online" is not available in response to a "passive offline" event.

1.59 Miami.guide/NODE_GUI_MODEM

Modem
=====

Init String
: The 'Init String' gadget

Exit String
: The 'Exit String' gadget

Dial prefix
: The 'Dial prefix' gadget

Dial suffix
: The 'Dial suffix' gadget

Null modem
: The 'Null modem' gadget

1.60 Miami.guide/NODE_GUI_MODEM_INIT

Init String

   The initialization string for your modem, usually set by MiamiInit.

1.61 Miami.guide/NODE_GUI_MODEM_EXIT

Exit String

   The string sent to your modem when Miami quits. Most users do not need this, but it can be useful if multiple programs share the modem port, and your modem needs to be reset to default settings before Miami exits.

1.62 Miami.guide/NODE_GUI_MODEM_PREFIX

```
Dial prefix
-----------
```

The command your modem uses for dialing, i.e. the string prepended to the phone number. This is usually "ATDT" or "ATDP".

1.63 Miami.guide/NODE_GUI_MODEM_SUFFIX

```
Dial suffix
-----------
```

The string that needs to be appended to your phone number to complete the dial command. This is usually "\r".

1.64 Miami.guide/NODE_GUI_MODEM_NULLMODEM

```
Null modem
----------
```

Miami usually assumes that you have a modem connected to your serial port. If your Amiga is directly connected to another computer using a null-modem cable, then you need to activate this gadget. It prevents any modem commands ("AT commands") from being sent, and Miami will not wait for any responses such as "OK" or "CONNECT".

With "null-modem" activated the meaning of the "Use CD" gadget on the "Interface" page changes:

* If your machine is connected to a computer that requires a login sequence to establish the SLIP/PPP link, then you should deactivate the "Use CD" gadget. Miami then uses the dial script specified in the "Dialer" window, but without dialing a number first. This option is most useful when connecting to a Unix or Linux box that runs a getty with login/password check on its serial port.

* If your machine is connected to a computer that runs its serial port in dedicated SLIP/PPP mode (e.g. another Amiga running Miami), then you should activate the "Use CD" gadget. Miami will then completely bypass any dial script and immediately proceed with the protocol negotiation.

1.65 Miami.guide/NODE_GUI_LOGGING

Logging
=======

1.66 Miami.guide/NODE_GUI_LOGGING_CONSOLE

Console

In this gadget you can specify the AmigaDOS stream name of the console window that Miami uses for system log messages. This file is kept open after the first system message has occured, so you should use the "CON:" modifiers "/AUTO/CLOSE" to be able to close the window without losing old system messages.

1.67 Miami.guide/NODE_GUI_LOGGING_FILE

File

In this gadget you can specify the AmigaDOS file name of the file where Miami stores system log messages. If the file already exists then Miami appends to this file, i.e. old file contents are not deleted.

1.68 Miami.guide/NODE_GUI_LOGGING_PHONE

Phonelog

Miami can log any online and offline events in order to assist in phone bill management.

The two "Phonebill" gadgets let you enable phone logging and specify the name of a file to which Miami appends billing records.

At the moment only ASCII format is supported, with records as follows:

```
Online: 27.07.1996 17:48:11 (5551234)
Passive offline: 27.07.1996 17:48:11
Active offline: 27.07.1996 17:48:11
Reconnect: 27.07.1996 17:48:11
```

The "Online" record contains the phone number that was dialed in "()". "Reconnect" occurs when Miami goes online without actually dialing, e.g. after rebooting the Amiga.

The difference between "passive" and "active" offline is that an "active" offline is voluntary, i.e. the result of an "OFFLINE" ARexx command, someone clicking on the "Offline" gadget etc. A "passive" offline is the result of your modem hanging up or your Internet provider disconnecting you.

1.69 Miami.guide/NODE_GUI_GUI

GUI
===

Quit requester
The 'Quit requester' gadgets

Offline requester
The 'Offline requester' gadget

Error requester
The 'Error requester' gadget

Dialer
The 'Dialer' gadgets

1.70 Miami.guide/NODE_GUI_GUI_REQQUIT

Quit requester

You can configure when Miami shall display a 'Quit requester':

* always

* when programs that use Miami are still running.

* when Miami is online

or combinations of the above.

1.71 Miami.guide/NODE_GUI_GUI_REQOFFLINE

Offline requester

If you activate this checkmark then Miami asks you before going offline.

1.72 Miami.guide/NODE_GUI_GUI_REQERRORS

Error requester

Normally Miami displays an error requester if any problems occur during dialing or while configuring the link. If you disable this checkmark then such errors are silently ignored, and Miami does not display an error requester.

1.73 Miami.guide/NODE_GUI_GUI_DIALER

Dialer

The standard dialer window has three parts: a help text at the top, several buttons in the middle, and a dialog window at the bottom. With the three "Dialer" checkmarks you can enable or disable each of these three parts.

If you disable the dialog window then the dialer will display a single line of text only, that contains the dialer command currently being executed.

1.74 Miami.guide/NODE_GUI_MISC

Misc
====

There are three more gadgets in Miami that are not described in any of the previous sections:

* "Online": Causes Miami to start dialing and try to go online.

* "Offline": Causes Miami to hang up the line and go offline.

* A listview gadget on the left side of the Miami window, that is used to select one of the configuration pages.

1.75 Miami.guide/NODE_DIALERLANG

```
Dialer Command Language
***********************

   The following commands are supported by the dialer:

ABORT "text1","text2",...
     Specify a list of texts that cause Miami to completely abort
     dialing, e.g. "NO DIALTONE" from the modem.

DELAY secs
     Wait for the specified number of seconds.

DIALNEXT "text1","text2",...
     Specify a list of texts that cause Miami to hang up the phone and
     dial the next number, e.g. "BUSY" from the modem.

PARSEPASSWORD "endchar"
     Parses all characters from the modem up to, but not including
     <endchar>, and replaces the current password by this text. This
     command can be useful for one-time password systems that send the
     password for the next session during login.

REDIAL "text1","text2",...
     Specify a list of texts that cause Miami to hang up the phone and
     redial the current number, e.g. "BUSY" from the modem.

SAVECONFIG
     Save the current configuration (settings) to disk. This command is
     usually used after PARSEPASSWORD to save the settings containing
     the new password.

SEND "text"
     Send <text> to the modem. A linefeed/carriage return is not
     automatically appended. Miami recognizes the following standard
     control sequences: \",\\,\r,\n. In addition "\u" and "\p"
     are supported to send the current login id (user id) or password,
     respectively.

SENDBREAK
     Send a serial port "break" signal. This is used by some terminal
     servers to switch to command mode.

SENDPASSWORD
     Send the current password, followed by a "\r".
```

```
SENDUSERID
     Send the current user id (login id), followed by a "\r".

TIMEOUT secs
     Specify the amount of time to wait for a text during WAIT or
     WAITPPP before giving up.

WAIT "text"
     Wait for "text" to be received from the modem.

WAITPPP
     Wait for the server to switch to PPP mode.
```

With the commands "ABORT", "DIAL" and "DIALNEXT" you can specify the keyword "TIMEOUT" (without the quotes), instead of a text in quotes, e.g.

```
     ABORT "NO CARRIER",TIMEOUT
```

This means that Miami will abort the dial script when a timeout occurs. Other options are to dial the current number again, or to dial the next number when a timeout occurs.

1.76 Miami.guide/NODE_AREXX

```
                ARexx Interface
***************
```

The name of the Miami ARexx port is "MIAMI.1". At the moment only the following commands are supported:

```
ISONLINE
     Checks if Miami is online end sets the error code ("RC")
     accordingly.  1 means: Miami is online. 0 means: Miami is offline.

ONLINE
     Attempt to go online. Same as clicking on the "Online" gadget.

OFFLINE
     Hang up and go offline. Same as clicking on the "Offline" gadget.

QUIT
     Quit Miami.

CHANGEDB
     Tells Miami to re-read the file "ENVARC:MiamiChangeDB" to update
     the settings. Please see
                Client settings
                 for more details how to
     use this feature.

HIDE
     Iconifies the Miami user interface.

SHOW
```

```
     Deiconifies the Miami user interface.

GETSETTINGSNAME
     Returns the file name of the current settings file in the result
     variable.
```

1.77 Miami.guide/NODE_EXCONFIG

```
                Exchanging Settings
*******************
```

The Miami settings are saved in an IFF file in a format that is currently intentionally undocumented. However Miami allows you to import and export settings in a variety of ways:

```
                Distribution format
                                        Importing/exporting settings for ↩
                                           distribution

                Exchanging passwords
                                       Exchanging password files

                Client settings
                                            Custom settings for some clients
```

1.78 Miami.guide/NODE_EXCONFIG_DIST

```
Distribution format
===================
```

Miami allows you to export settings into an ASCII format that is suitable for distribution, e.g. to upload it to Aminet, or to give it to other users who have accounts with the same Internet provider. It can also be used by Internet providers to preconfigure complete Miami settings for new user.

The ASCII file format contains a header, followed by a variable number of parameters.

When exporting files Miami only includes those parameters that are related to the provider, but not those that are related to the individual user's system setup or that are security-relevant in any way. This means you can safely export your settings and give the file to other user, without compromising sensitive information like passwords.

When importing files Miami does support user-related information like passwords though, so providers can write Installer scripts which ask the user for his login id and password, and which then create an ASCII settings file for Miami that contains all information required by Miami.

To get an idea how the ASCII file looks just export your current settings to ASCII. The general format is

* a 2-line header. Each line starts with a "$" sign. Do not modify this header.

* a variable number of lines starting with a ";". These lines are comments and can be edited freely.

* a variable number of lines that specify parameters.

Most parameters are specified in a single line. These lines look like this:

```
PARAMETER=value
```

Some parameters (e.g. the dial script) require several lines. In this case the format is as follows:

```
PARAMETER=%
first value
second value
third value
%
```

This means a single "%" indicates a multi-line parameter, and another "%" as the only character on a line indicates the end of the list of values.

The order of parameters within the file is arbitrary. You should not make any assumptions that Miami stores parameters in a specific order.

List of supported parameters: A (m) indicates a multi-line parameter. A (i) indicates that the parameter is only imported, but never exported. "(m)" and "(i)" are not actually part of the ASCII file.

Values indicated as "A / B" means that the value is a single character, either "A" or "B".

```
DEVNAME= (i)
     devicename

UNIT= (i)
     device unit number

BAUD= (i)
     serial port speed

PROTOCOL=
     P / S    (ppp or slip)
```

```
FLOWCONTROL= (i)
     H / S    (hardware (RTS/CTS) or software (Xon/Xoff) handshaking)

EOFMODE= (i)
     Y / N / A    (yes / no / auto)

SERMODE=
     8N1 / 7E1 / 7O1

MTU=
     integer

IPTYPE=
     D / S    (dynamic or static)

IP=
     1.2.3.4

CD= (i)
     Y / N    (Use CD)

BOOTP=
     Y / N    (Use BootP)

INACTIVITY=
     N / I / P    (inactivity type: none, ICMP, PPP)

INACTIVITYDELAY=
     minutes

PAPNAME= (i)
     username

PAPPWD= (i)
     password

PAPSAME=
     Y / N

ACCM=
     000a0000

VJC=
     Y / N

QUICKRECONNECT=
     Y / N

DIALNAME= (i)
     login id

DIALPWD= (i)
     password

INITSTRING= (i)
     modem_init_string
```

```
DIALPREFIX= (i)
     dial_prefix

DIALSUFFIX= (i)
     dial_suffix

DIALSCRIPT= (m)
     dial_script

DIALNUMBERS= (i)(m)
     phone_numbers

DIALMAXREPEAT=
     maxrepeat

DIALREPEATDELAY=
     repeatdelay

HOSTDYNAMIC=
     Y / N    (host name dynamic: yes / no)

HOSTNAME= (i)
     hostname

REALNAME= (i)
     real_name

USERNAME= (i)
     user_name

DOICMP=
     Y / N

FAKEIP=
     Y / N

DBHOSTS= (m)
     host_database

DBNETWORKS= (m)
     network_database

DBDOMAINS= (m)
     domain_database

DBDNSSERVERS= (m)
     dns_servers_database
```

1.79 Miami.guide/NODE_EXCONFIG_PASSWORDS

```
Exchanging passwords
====================

   Miami allows you to freely import and export all files from the
```

Unix/AmiTCP db directories, with one exception: the passwd file can be imported, but the passwords are cleared in the process, and thus have to be reentered manually in Miami.

The reason for this is: AmiTCP (at least up to version 4.3) uses the DES algorithm for password encryption. DES is a cryptographically strong encryption algorithm that is subject to US export restrictions. A program implementing DES may not be exported from the US without an individual permit, and the US government currently does not issue such permits.

The result is that any kind of export of AmiTCP from the US is illegal. This includes downloading the AmiTCP archive from an ftp server in the US to a computer outside of the US. For this reason AmiTCP may not be uploaded to all Aminet sites, severely restricting the availability of AmiTCP.

For Miami things would have been even worse: since I am developing Miami within the US (not in Finland like NSDi) I would not have been allowed to send Miami to anybody outside of the US, regardless of the way I distribute it. I therefore decided not to use DES in Miami, but to use a different encryption algorithm that is not subject to US export restrictions.

Miami uses an iterated version of MD5 for password encryption. This algorithm is cryptographically strong, i.e. not known to be breakable except by exhaustive search, just like DES. However since MD5 is, unlike DES, a one-way algorithm, it cannot be decrypted and therefore is not subject to US export restrictions.

This means it is completely legal to import and export Miami to and from the US, to upload Miami to Aminet sites and other ftp sites, and to use Miami in the US and other countries (unless some country forbids the use of MD5).

I am sorry for the problems this may cause for users who have to maintain multiple and/or large password files, but I do not see any other way of handling this situation.

1.80 Miami.guide/NODE_EXCONFIG_CLIENTS

Custom client settings
======================

Some TCP/IP clients such as AmiTalk require changes to the settings database that most protocol stacks store in the "db" directory. Usually entries have to be added to the "services" or "inetd.conf" file.

With Miami you can make the appropriate changes directly through the graphical user interface, i.e. just select the "Database" page, the correct section (e.g. "services"), and add the entries you need.

In some situations it can be more convenient to automatize this process, e.g. to have the Installer script of a TCP/IP client make the

required changes by itself, without bothering the user. With Miami this works as follows:

* You first need to append a line to the file "ENVARC:MiamiChangeDB" that looks as follows:
  ADD services ntalk 518/udp
  or
  ADD inetd ntalk dgram udp wait root Servers:talkd (talkd)
  Whenever Miami is started it automatically reads the contents of this file (if it exists), updates the settings, and saves the resulting settings.

* If Miami is running when the client is installed and you want Miami to update its settings immediately you should send the "CHANGEDB" ARexx command to Miami after modifying the above file.

To summarize: In your Installer scripts you should have statements as follows to automatically configure Miami for your client:

```
echo >>ENVARC:MiamiChangeDB "ADD services ntalk 518/udp"
rx "address MIAMI.1;CHANGEDB"
```

If Miami is running it updates the settings immediately. Otherwise Miami picks up the changes the next time it is started.

1.81 Miami.guide/NODE_UTILITY

Utility Programs
****

MiamiNetStat
MiamiNetStat

Other utility programs are planned for future versions, but not available yet.

1.82 Miami.guide/NODE_UTILITY_NETSTAT

MiamiNetStat
============

MiamiNetStat is a tool to display configuration parameters and statistics. It is almost identical in functionality to the version of "netstat" that is included with 4.4BSD, but has some additional functions to display link-level statistics.

Usage:

* MiamiNetStat [-Aan] [-f address_family]

* MiamiNetStat [-dimnrs] [-f address_family]

* MiamiNetStat [-dn] [-] [-I interface]

* MiamiNetStat [-s] [-] [-L interface]

* MiamiNetStat [-p protocol]

The MiamiNetStat command symbolically displays the contents of various network-related data structures. There are a number of output formats, depending on the options for the information presented.

The first form of the command displays a list of active sockets for each protocol.

The second form presents the contents of one of the other network data structures according to the option selected.

Using the third form MiamiNetStat will display information regarding packet traffic on the specified network interface.

The fourth form displays link-level configuration information or (with the "-s" flag) link-level statistics for the specified network interface.

The fifth form displays statistics about the named protocol.

The options have the following meaning:

-A
With the default display, show the address of any protocol control blocks associated with sockets; used for debugging.

-a
With the default display, show the state of all sockets; normally sockets used by server processes are not shown.

-d
With an interface display (option i or I), show the number of dropped packets.

-f address_family
Limit statistics or address control block reports to those of the specified address family. Only the address family "inet" is currently recongized.

-I interface
Show information about the specified interface.

-i
Show the state of interfaces which have been configured.

-m
Show statistics recorded by the memory management routines (the network manages a private pool of memory buffers).

-n
     Show network addresses as numbers (normally MiamiNetstat interprets addresses and attempts to display them symbolically). This option may be used with any of the display formats.

-p protocol
     Show statistics about the specified protocol, which is either a well-known name for a protocol or an alias for it. A null response typically means that there are no interesting numbers to report. The program will complain if the protocol is unknown or if there is no statistics routine for it.

-s
     Show per-protocol statistics. If this option is repeated, counters with a value of zero are suppressed.

-r
     Show the routing tables. When "-s" is also present, show routing statistics instead.

The default display, for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form "host.port" or "network.port" if a socket's address specifies a network but no specific host address. When known the host and network addresses are displayed symbolically according to the "hosts" and "networks" databases. If a symbolic name for an address is unknown, or if the "-n" option is specified, the address is printed numerically, according to the address family.

The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit ("mtu") are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows a collection of information about the route stored as binary choices.

1
     RTF_PROTO1 Protocol specific routing flag #1

2
     RTF_PROTO2 Protocol specific routing flag #2

C
     RTF_CLONING Generate new routes on use

D
     RTF_DYNAMIC Created dynamically (by redirect)

G
     RTF_GATEWAY Destination requires forwarding by intermediary

H

```
     RTF_HOST Host entry (net otherwise)

L
     RTF_LLINFO Valid protocol to link address translation.

M
     RTF_MODIFIED Modified dynamically (by redirect)

R
     RTF_REJECT Host or net unreachable

S
     RTF_STATIC Manually added

U
     RTF_UP Route usable

X
     RTF_XRESOLVE External daemon translates proto to link address
```

Direct routes are created for each interface attached to the local host; the gateway field for such entries shows the address of the outgoing interface. The refcnt field gives the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The use field provides a count of the number of packets sent using that route. The interface entry indicates the network interface utilized for the route.

With the option "-L" MiamiNetStat displays link-level configuration information, such as the current state of the IPCP or LCP subprotocols of PPP, for the specified interface.

With the option combination "-sL" MiamiNetstat displays link-level statistics, including information about different types of packets, and checksum errors, for the specified interface.

Currently Miami only supports two interfaces:

```
lo0
     The local loopback interface

mi0
     The PPP/(C)SLIP interface using the interface driver built into
     Miami.
```

1.83 Miami.guide/NODE_COMPATIBILITY

```
Compatibility
*************
```

So far Miami has worked with all AmiTCP clients and servers it has been tested with, with one exception:

```
   The AmiTCP 4.x version of "telnet" does not normally work with Miami.
This is because that version of "telnet" uses some non-documented
features of "TCP:" that cannot be emulated by Miami.

   There are two solutions to this:

   * Use a different version of telnet, e.g. the version available from
     Aminet in comm/tcp, a terminal program together with
     telser.device, or "napsaterm" in telnet-mode. A graphical telnet
     client that works well with Miami is expected to be available soon.

   * Install the version of "inet-handler" that comes with AmiTCP
     4.0demo, create an appropriate mountlist entry for "TCP:", and type
     "mount TCP:" before starting Miami. "telnet" will then use the
     AmiTCP version of "TCP:" (still accessing the Miami TCP/IP stack,
     of course) instead of the version of "TCP:" built in to Miami.
```

1.84 Miami.guide/NODE_RESTRICTIONS

```
Restrictions
************

   The demo version has the following limitations:

   * After 60 minutes the modem hangs up the line.

   * The "Events" options "auto-online after passive offline" and
     launching ARexx scripts are not available.

   * The number of phone numbers in the dialer is limited to three.

   * Phone logging is disabled.

   * Some of the GUI customization options are disabled.
```

1.85 Miami.guide/NODE_HISTORY

```
History
*******

Version 1.1
     fifth public release version
        * Probably fixed the bug that sometimes the ARexx command
          "QUIT" did not work when the GUI is iconified.

        * When the dial dialog was disabled the dial text field would
          not always clip text properly - fixed.

        * Added several modem entries to MiamiInit.
```

* Corrected some MUI background patterns in Miami.

* Added a border of one additional pixel around the text field in the dial window to accomodate some fonts that do not correctly define their spacing.

* Fixed a slight rendering bug in the dial window that occured when the dial dialog was switched off.

* MiamiInit now attempts to find the correct MTU and passes it to Miami.

* Fixed yet another memory leak.

* Changed the appearance of the "General" page for the registered version.

* Launching MiamiRegister from Miami now works if Miami was started from the shell with a non-default path.

* Two slight performance enhancements in the TCP/IP kernel.

* Added warning message if someone tries to load an old settings file with an incomplete database (as generated by Miami 1.0.1 and 1.0.2).

Version 1.0.3
fourth public release version

* Change in the DNS resolver: Previously if the host name did not end in a '.' DNS lookups would first assume that the host name is an abbreviation and append all domain names one by one for DNS lookup, before doing a lookup on the host name alone. Now, after the change, any host name that contains at least one '.' is considered fully qualified, and DNS lookups are done for this host name first (without appending a domain). Only if these lookups fail, and if the host name does not end in a '.' are domain-based lookups done. This new strategy is different from what BSD usually does, but it apparently gives better performance (reduces the number of DNS lookups) if host name abbreviations are not used very often, and if DNS lookups can take a non-negligible amount of time for at least one predefined domain. The bottom line is: The "looking up host name" phase of web browsers should be quicker now than before, especially if you have more than one domain listed in Miami's domain database.

* Fixed a minor bug in the way the "daytime" service handles ENV:TZ.

* Added a GUI configuration page to customize many aspects of the user interface.

* The "quit" requester is now more customizable and useful, an "offline" requester has been added, and all protocol-related error requesters can be suppressed.

* The help text and buttons in the dial window can now be separately disabled, and it is possible to only switch to a much smaller dial window that just displays the currently executed command, not the complete dial dialog.

* If gethostbyname() failed incorrect error values were returned in errno and herrno.

* Added an "escape" function to PPP to allow other characters than 0-31 and 128-159 to be sent escaped.

* Executing the ARexx command "QUIT" from within an ARexx script launched from Miami works now, and quits the program after all ARexx scripts have returned.

* Added requester to Miami, MiamiInit and MiamiRegister to inform the user that MUI 3.3 or higher is required.

* syslog() did not work properly with all programs.

* syslog() now logs the process id if requested by a program.

* If Miami was started without a settings file the database was not initialized correctly.

* Added support for "dos" type servers in built-in InetD.

* The main window now acts as an AppWindow, i.e. if you drop the icon of a settings file on it Miami loads that settings file.

* If Miami is started from Workbench it now tries to use "ENV:Sys/def_MiamiApp.info" as its AppIcon. If this icon does not exist or if Miami was started from the Shell the standard built-in icon is used.

* There was one more special case left in which the memory allocation problem of version 1.0 still occured - fixed.

Version 1.0.2
third public release version

* Minor changes to MiamiRegister only, to reflect the new host name of the Miami registration server.

Version 1.0.1
second public release version

* Workaround for a problem with some Annex terminal servers.

* Control sequences "\p" and "\u" incorrectly added a "\r" character.

* The "passive offline" ARexx script is now executed before attempting to reconnect.

* There was some debugging output left in the code when Miami replied to an ARexx message - fixed.

```
        * PPP now no longer signals a "down" event to Miami when IPCP
          goes down, but only when LCP goes down. This should prevent
          problems with very slow terminal servers that try to
          renegotiate IPCP after it has already come up once.

        * Added ARexx commands "SHOW", "HIDE" and "GETSETTINGSNAME".

        * Added PPP option "Get DNS from IPCP" to disable IPCP-based
          DNS discovery. Some buggy PPP servers are unable to handle or
          reject IPCP DNS extensions correctly.

        * Miami now explicitly checks for MUI >=3.3.

        * Fixed a race condition in the unregistered version that could
          cause Miami to reallocate the same buffer over and over again
          after one hour of inactivity, quickly exhausting all
          available RAM.

        * Fixed a menu shortcut collision.

        * If ToolTypes were used to import MiamiInit settings or an
          exported configuration then the database was not initialized
          properly - fixed.

        * Some combinations of Slirp/TIA with Slip/PPP were not
          configured correctly after importing settings from MiamiInit.

        * Some minor changes in MiamiInit.

        * Fixed a memory leak in built-in InetD.

Version 1.0
     first public release version
```

1.86 Miami.guide/NODE_FUTURE

```
The future
**********

   more ARexx commands, utility programs, T/TCP, more ToolTypes,
support for single-device SANA-II setups, support for ISDN (if CAPI 2.0
really becomes available as promised),...  ...you name it.
```

1.87 Miami.guide/NODE_SUPPORT

```
Support
*******

   There are several ways to get technical support, updates etc.:
```

email
     kruse@nordicglobal.com

snail mail
     Holger Kruse
     12006 Coed Drive
     Orlando FL 32826
     USA

WWW
     http://www.nordicglobal.com/Miami.html

mailing list
     send "SUBSCRIBE miami-ml" in the body of a mail to
     "amiga-lists@nordicglobal.com".

1.88 Miami.guide/NODE_ACKNOWLEDGEMENTS

Acknowledgements
****

   My sincere thanks go to

   * the early alpha and beta testers Karl Bellve, Mike Fitzgerald,
     Adam Hough, Daniel Saxer, Stefan Stuntz and Oliver Wagner.

   * Karl Bellve and Daniel Saxer for their great support efforts.

   * NSDi for the first publically available TCP/IP protocol suite for
     AmigaOS and its very usable API.

   * James Cooper, Steve Krueger and Doug Walker for the SAS/C
     development system and their great support.

   * Stefan Stuntz for his nice graphical user interface package MUI.

   * Klaus Melchior for his MUI custom class "Busy.mcc".

   * Robert Reiswig for loaning me some important computer equipment.

   * the University of California for their successful continued work on
     the excellent BSD networking code.

   * Reinhard Spisser and Sebastiano Vigna for their Amiga port of
     "makeinfo".

   * Paul Trauth, the winner of the Miami logo contest, for his nice
     collection of images.

   * John Pszeniczny for his nice variations of the "Miami" logo.

   * all users who decide to register Miami.