Miami

COLLABORATORS			
	TITLE : Miami		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 10, 2022	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Miami

1.1 Miami.guide

```
                Miami
*****

   This is the documentation for Miami V3.2b, an integrated TCP/IP
system for AmigaOS. Copyright (C) 1996-1998 Nordic Global Inc.  All
rights reserved. Program and documentation by Holger Kruse.
```

1.2 Miami.guide/NODE_DISCLAIMER

```
                Disclaimer
**********

Miami IS SUPPOSED TO BE A TCP/IP PACKAGE FOR AmigaOS THAT CAN BE USED
TO CONNECT YOUR AMIGA TO THE INTERNET BY MODEM OR THROUGH A NETWORK
DEVICE.  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 THOSE
```

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

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.

Some of Miami's GUI modules rely on Magic User Interface (MUI). MUI is Copyright by Stefan Stuntz.

Some of Miami's GUI modules require 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

Some of Miami's GUI modules require gtlayout.library by Olaf Barthel. Here is the associated copyright notice:

Begin Quote

Copyright © 1993-1996 by Olaf ‘Olsen’ Barthel Freely distributable.

End Quote

Some of Miami's GUI modules use images based on MagicWB, which have been copied with permission from the author. Here is the associated copyright notice. More information is available from the file ‘MagicWB.readme’ in the Miami distribution.

Begin Quote

Copyright © 1992-97, Martin Huttenloher

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 !

Key files are non-transferable and may not be sold or traded to any other person or organization. They are intended to be used only by the person who registered.

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.

Please contact me at kruse@nordicglobal.com if for some reason you cannot run the registration program MiamiRegister.

The registration fee is US$ 35 for a standard, ‘full’ Miami license. 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.

Users who already registered Miami 2.x will need new keyfiles for Miami 3.x. The upgrade policy is:

* If you obtained your Miami keyfiles after June 15th, 1997, then the upgrade to Miami 3.x is free.

* If you obtained your Miami keyfiles before June 15th, 1997, then you will need an upgrade code to upgrade your keyfiles. Upgrade codes are available from Nordic Global Inc. (for US$ 12.00) and from some registration sites.

To upgrade your keyfiles please run MiamiRegister. The program will offer you upgrade options and will also tell you whether there is an upgrade charge for you.

If you already have keyfiles for 3.0 then you do not need to upgrade again for 3.2. Keyfiles for 3.0 work with 3.2 without change.

1.5 Miami.guide/NODE_INTRODUCTION

Introduction
****

Miami is an integrated TCP/IP system for AmigaOS, that allows you to access the Internet or a local-area network by modem or by some other network device (e.g. Ethernet) in a very easy way.

Miami is based on the latest version (4.4BSD V2) of the official BSD networking code, plus some of the extensions made by third parties (such as FreeBSD T/TCP and Path MTU discovery code). This means 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, an interface to SANA-II drivers, a graphical user interface for program control and configuration, a client for SOCKS proxy servers and many other features.

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 has very extensive support for modem-based dial-up connections 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 can also be used with a non-modem connection, e.g. an Ethernet interface, an Arcnet interface, or a cable modem.

Miami supports several different GUI modules for its configuration. When controlling Miami (going online or offline, or changing settings for instance) a GUI module has to be loaded. Once Miami is online it is possible to unload the GUI module in order to save memory. You can reload the GUI module at any time if you want to make any changes to your setup.

Miami currently supports the following GUI modules:

MUI
: This module requires MUI (Magic User Interface) 3.8 or higher, and generates a user interface in the typical MUI style.

MUIMWB
: This module is identical to the 'MUI' module, except that it also uses images in some places, not just text, and was more elaborately designed. It requires MUI 3.8 or higher, and a screen with at least 8 colors using the MagicWB color palette.

GTLayout
: This module generates a GadTools-based user interface, and requires Olaf Barthel's gtlayout.library version 40 or higher. It does not require MUI.

The recommended GUI module is 'MUI' or (for deep screens) 'MUIMWB'. The GTLayout module can be used as well, but some of the functions of the MUI module are not accessible through it (e.g. drag&drop sorting of database entries).

Before starting 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, netmask and all other configuration variables.

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

If you want to use Miami with a local Ethernet connection then you may want to configure Miami manually, without MiamiInit. You can use MiamiInit for that, but MiamiInit currently does not support the new MNI drivers. If you use MiamiInit then you can only use SANA-II drivers with your Ethernet board. This will be changed in a future version.

1.6 Miami.guide/NODE_REQUIREMENTS

Requirements
****

To use Miami you need:

* an Amiga running OS 2.04 or higher

* MUI 3.8 or higher if you want to use one of the MUI modules, or alternatively gtlayout.library V40 or higher for the GTLayout module.

You will also need some hardware for networking and a machine to connect to. This could for instance be:

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

* an Ethernet board, a cable modem, and a SLIP/PPP account as described above.

* an Ethernet board connecting your machine to a local area network.

Note that Miami does not require ppp.device, appp.device, amippp.device or (rh)(c)slip.device. The protocols PPP and (C)SLIP are built into Miami, in versions more efficient and more advanced than those currently available in SANA-II devices.

1.7 Miami.guide/NODE_INSTALLATION

```
Installation
************

   Miami is packaged in the following archives:

Miami32b-main.lha
     The main archive. Everyone needs this.

Miami32b-000.lha
     The 68000/010 version of Miami. You need this if your Amiga has a
     68000 or 68010 CPU.

Miami32b-020.lha
     The 68020+ version of Miami. You need this if your Amiga has a
     68020, 68030, 68040 or 68060 CPU.

Miami32b-MUI.lha
     The MUI module for Miami. You need this if you want to use Miami
     together with MUI >=3.8.

Miami32b-GTL.lha
     The GTLayout module for Miami. You need this if you want to use
     Miami together with gtlayout.library V40 or higher.

   Everybody needs to download the main archive, one of the two CPU
archives (000 or 020), and at least one of the two GUI module archives.
You may install more than one GUI module, if you like.

   Download all archives, unarchive them into the same (temporary)
directory, and then execute the Installer script in that directory to
install Miami. The Installer script can be used for a new installation
or for updates.

   All files are copied from the installation directory to a single
target directory, and no system files or system directories are
touched, with one exception:

   The Installer script asks you whether you want to create a "Miami:"
assign, and then adds the required statements to your user-startup
file. Doing this is required.  If you skip this step during the
installation then you must manually create the assign before starting
Miami. Otherwise Miami will not work properly.
```

1.8 Miami.guide/NODE_MIAMIINIT

```
MiamiInit
*********

   MiamiInit is a utility program that tries to determine all
configuration parameters for Miami that are required for a serial
connection (SLIP or PPP) or a SANA-II connection (Ethernet, Arcnet
etc.), and then saves a configuration file that can later be used by
Miami.
```

The first thing you should do to configure Miami after installation is to run MiamiInit, and go through the dialog. In the process MiamiInit connects to your network 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.

Note that depending on the configuration of your network MiamiInit might not be able to find all information entirely by itself. It is not an error if MiamiInit asks you for things like IP addresses or netmask during the configuration. This just means that there is no server on the network which provides this kind of information to MiamiInit. In this case you need to ask your Internet provider or network administrator for the missing information.

Generally, if you are setting up a very small local network, that just consists of Amigas and PCs, then you will usually have to enter most of the information yourself. On the other hand if you are connecting to the Internet or to an existing corporate network which has been set up to configure new machines then MiamiInit can often find most or all of the information from a server.

MiamiInit currently does not support MNI drivers. If you want to configure Miami for Ethernet then you either need to use MiamiInit and configure it for SANA-II, or configure Miami manually for MNI. Of course you can also first run MiamiInit, configure the system for SANA-II, and then later switch to MNI, after importing the settings into Miami.

1.9 Miami.guide/NODE_TOOLTYPES

ToolTypes
****

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

PACKETDEBUG
: Initiates packet-level debugging mode. If you specify "PACKETDEBUG=10" or "PACKETDEBUG=20" then Miami creates a file "Miami.debug" with a hex dump of all sent and received packets. You should only use this during debugging, not during normal operation, because these logs grow very quickly and consume a lot of CPU time. A value of 10 logs packet payloads only. A value of 20 also logs raw packet data (for PPP/SLIP).

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.

AREXX
     The argument "AREXX=filename" tells Miami to execute the specified
     ARexx script upon startup.

PUBSCREEN
     The argument "PUBSCREEN=name" sets the public screen you want Miami
     to open on. Note that the MUI modules have their own method of
     configuring screens, through MUI.

GUI
     The argument "GUI=name" tells Miami which GUI engine to use for
     the user interface. This overrides any user interface choise in the
     settings file.

NOGUI
     The argument "NOGUI" causes Miami to start up without showing the
     user interface.

   DO NOT attempt to use undocumented ToolTypes ! Such ToolTypes usually
do not do what you expect them to do, and might reduce the compatibility
or performance of Miami.
```

1.10 Miami.guide/NODE_MENUS

```
Menus
*****

   Description of all menu items:

Project/About...
     Show information about Miami.
```

Project/About MUI...
: Show information about MUI (Magic User Interface). This menu item is only available when using one of the MUI user interface modules.

Project/Iconify
: Iconify all windows of Miami. Note that for some interface modules (e.g. GTLayout) this is identical to 'Project/Kill GUI'.

Project/Kill GUI
: Iconify all windows of Miami and unload the GUI module from memory.

Project/Offline without hangup
: Go offline without hanging up the modem line first.

Project/Quit without hangup...
: Leave Miami without hanging up the modem line first.

Project/Quit...
: Leave Miami.

Settings/Load...
: Load a settings file.

Settings/Save
: Save the current settings into the current settings file.

Settings/Save as...
: Save the current settings into a new settings file.

Settings/Save as default
: Save the current settings as the default for Miami.

Settings/Create icon
: Create a project icon for each settings file saved.

Settings/Import from distribution...
: Import a settings file from Miami's distribution (ASCII) format.

Settings/Export from distribution...
: Export the settings into a file in Miami's distribution (ASCII) format.

Settings/Import from MiamiInit V2...
: Import a settings file from MiamiInit version 2. This function is obsolete. You should use MiamiInit version 3 and the import function 'Settings/Import from MiamiInit V3...' instead.

Settings/Import from MiamiInitSANA2 V2...
: Import a settings file from MiamiInitSANA2 version 2. This function is obsolete. You should use MiamiInit version 3 and the import function 'Settings/Import from MiamiInit V3...' instead.

Settings/Import from MiamiInit V3...
: Import a settings file from MiamiInit version 3.

Settings/MUI Settings...
: Open the MUI configuration window. This menu item is only

```
     available when using one of the MUI user interface modules.
```

1.11 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:
```

1.12 Miami.guide/NODE_GUI_GENERAL

```
                General
=======

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

   With some GUI modules (e.g. MUI) this page is selected by clicking
on "General" in the Listview. With other GUI modules the Miami main
window always shows the contents of the "General" page, and other pages
pop up in subwindows.
```

Register	The ‘Register’ gadget

1.13 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.
```

1.14 Miami.guide/NODE_GUI_INTERFACE

```
                Interface
=========
```

Interface type	The ‘Interface type’ gadget
Driver / Unit	The ‘Driver’ and ‘Unit’ gadgets
Speed	The ‘Speed’ gadget
Use CD	

```
                                        The ‘CD’ gadget

Protocol
                                       The ‘Protocol’ gadget

Flow control
                                   The ‘Flow control’ gadget

EOF mode
                                       The ‘EOF mode’ gadget

Serial mode
                                    The ‘Serial mode’ gadget

MNI Options
                                    The ‘MNI Options’ gadget

IP type / address
                              The ‘IP’ gadgets

Netmask type / address
                         The ‘Netmask’ gadgets

Gateway type / address
                         The ‘Gateway’ gadgets

Multicasts
                                     The ‘Multicasts’ gadget

Mapping
                                        The ‘Mapping’ gadget

MTU
                                            The ‘MTU’ gadget

SANA-II parameters
                             The ‘SANA-II parameters’ gadget

MNI parameters
                                 The ‘MNI parameters’ gadget

MNI information
                                The ‘MNI information’ gadget

Find boards
                                    The ‘Find boards’ gadget

Inactivity
                                     The ‘Inactivity’ gadgets
```

1.15 Miami.guide/NODE_GUI_INTERFACE_TYPE

```
Interface type
--------------

   This gadget selects the type of interface you want to use. The exact
layout of the ‘Interface’ page depends on the type you choose, i.e. only
those gadgets that are applicable for the type of interface you chose
are shown.

   Available types:

builtin serial (PPP/CSLIP)
     The standard built-in implementation of PPP or (C)SLIP, running on
     top of Miami own serial driver for the Amiga’s built-in serial
     port.  This interface type can be used with the Amiga’s built-in
     serial port only, and does not require additional drivers. Usually
     using this interface type provides better performance than using
     the type ‘serial driver’ with serial.device (or a replacement
     device).

serial driver (PPP/CSLIP/IP)
     The standard built-in implementation of PPP or (C)SLIP, running on
     top of any serial.device-compatible device. This interface type
     does not require a SANA-II driver.

SANA-II point-to-point
     A SANA-II driver for a device that connects exactly two machines,
     like SLIP, PPP or PLIP.

SANA-II Ethernet
     A SANA-II driver for an Ethernet device, like the A2065 board, the
     Hydra board or the Ariadne board.

SANA-II standard "old" Arcnet
     A SANA-II driver for an Arcnet board, like the A2060 board. This
     setting uses the standard "old" RFC1051 Arcnet encapsulation,
     which is more popular on Amiga networks than the "new" RFC1201
     encapsulation.  Use the "old" encapsulation when you need to
     connect your Amiga to AmiTCP/IP, Inet-225 or NetBSD 1.1.

SANA-II "new" Arcnet
     A SANA-II driver for an Arcnet board, like the A2060 board. This
     setting uses the "new" RFC1201 encapsulation. It does not
     interoperate with AmiTCP/IP or NetBSD 1.1, but you might need this
     "new" standard if you want to connect your machine to other
     platforms such as Windows 95.

SANA-II other bus/ring
     A SANA-II driver for a bus or ring device other than Ethernet or
     Arcnet. Choose this setting if your hardware can connect more than
     two machines, but is neither Ethernet-compatible nor
     Arcnet-compatible.
```

1.16 Miami.guide/NODE_GUI_INTERFACE_DEVICE

```
Driver / Unit
-------------

For serial interfaces:
     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’ or ‘artser.device’ ‘0’. 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’
     and ‘v34serial.device’.

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

For SANA-II interfaces:
     Enter the device name and unit number of your SANA-II hardware.
     The unit number is 0 in most cases.

For MNI interfaces:
     Enter the driver name and unit number of your MNI hardware. If you
     do not know the correct unit number then only enter the driver
     name, click on "Find Boards", select your board in the window that
     pops up, and click on "OK". This sets the unit number to the
     correct value.
```

1.17 Miami.guide/NODE_GUI_INTERFACE_SPEED

```
Speed
-----

   (This option is available for serial devices only.)

   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.
```

1.18 Miami.guide/NODE_GUI_INTERFACE_CD

```
Use CD
------

   (This option is available for serial devices only.)
```

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") and then save the modem settings to NV-RAM from a terminal program (usually "AT&W"), 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 not dial a number, i.e. the "ATDT..." command is skipped, and the list of phone numbers is meaningless.

1.19 Miami.guide/NODE_GUI_INTERFACE_PROTOCOL

Protocol

(This option is available for serial devices only.)

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

1.20 Miami.guide/NODE_GUI_INTERFACE_FLOW

Flow control

(This option is available for serial devices using external serial drivers only. The builtin serial driver always uses RTS/CTS.)

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.

1.21 Miami.guide/NODE_GUI_INTERFACE_EOF

EOF mode

(This option is available for serial devices using external serial drivers only. The builtin serial driver always has EOF-mode enabled.)

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 then you can manually override the default setting by choosing "on" or "off".

1.22 Miami.guide/NODE_GUI_INTERFACE_SERIAL

Serial mode

(This option is available for serial devices using external serial drivers only. The builtin serial driver always uses 8N1.)

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 dial-in 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.

1.23 Miami.guide/NODE_GUI_INTERFACE_MNIOPT

```
                MNI Options
-----------

   (This option is available for MNI drivers only.)

   The gadget "MNI Options" allows you to enter parameters to configure
the behavior of the MNI driver in more detail. The types of options you
can use vary with each driver. Please check
                MNI drivers
                 for more
information on MNI drivers and supported options.
```

1.24 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.

For serial interfaces:
     If your Amiga has a fixed IP address choose "static" and enter the
     IP address your provider told you. If your provider assigns you a
     dynamic IP address for each connection 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.

For SANA-II point-to-point interfaces:
     If your machine has a fixed address then choose "static" and enter
     the IP address. If the address is assigned by a local BootP/DHCP
     server then choose "DHCP". If the SANA-II device determines the
     dynamic address by itself (e.g. ppp.device) then choose "SANA-II'.

For SANA-II bus/ring interfaces:
     If your machine has a fixed address then choose "static" and enter
     the IP address. If the address is assigned by a local BootP/DHCP
     server then choose "DHCP". If the address is assigned by a local
     RArp server then choose "RArp".
```

1.25 Miami.guide/NODE_GUI_INTERFACE_MASK

```
Netmask type / address
----------------------
```

(This option is available for SANA-II bus/ring devices only.)

Your netmask needs to be configured correctly so that Miami knows how many machines are in your local network. There are three ways of setting the netmask:

static
: Ask your network administrator for the correct netmask and enter it.

DHCP
: Miami tries to get the correct netmask from a local BootP/DHCP server.

ICMP
: Miami tries to get the correct netmask from a local server that supports ICMP netmask discovery.

1.26 Miami.guide/NODE_GUI_INTERFACE_GWAY

Gateway type / address

(This option is available for SANA-II bus/ring devices only.)

Your default gateway needs to be configured correctly so that Miami knows where to send packets that are not intended for a machine on your local network. There are three ways of setting the gateway:

static
: Ask your network administrator for the correct gateway and enter it.

DHCP
: Miami tries to get the correct gateway from a local BootP/DHCP server.

ICMP
: Miami tries to get the correct gateway from a local server that supports ICMP router discovery.

1.27 Miami.guide/NODE_GUI_INTERFACE_MULTICASTS

Multicasts

(This option is available in the registered version only.)

Miami support Level-2 multicasting, i.e. both sending, and receiving multicast messages.

If you want to use applications that need support multicasting (none are available yet), you might have to enable Multicasts in Miami. The possible settings are:

disabled
: Multicasts are disabled.

send as broadcasts
: Multicasts are sent as link-level broadcasts (or for point-to-point devices: as ordinary packets).

send as multicasts
: Multicasts are sent as link-level multicasts. This option is only available for Ethernet boards.

Note: Multicasts should only be enabled for an interface if you receive your multicast feed directly from this interface. If you get your multicast feed through a tunnel using MiamiMRouteD then you usually need to disable multicasts on Miami’s interface, because MiamiMRouteD handles multicasting by itself.

1.28 Miami.guide/NODE_GUI_INTERFACE_MAPPING

Mapping

(This option is available for SANA-II Arcnet devices only.)

Arcnet supports two different standards to map IP addresses to hardware addresses:

Arp
: Arp (Address resolution protocol) is used. This is the recommended default, and is also what AmiTCP/IP uses.

direct
: The least-significant 8 bits of the IP address are mapped to the hardware address. This is what NetBSD 1.1 uses.

If you have at least one NetBSD 1.1 machine in your Arcnet network then you can make life easier for you by choosing "direct" mapping instead of creating manual Arp entries on all machines.

In all other cases you should choose "Arp" on all machines. Newer ("current") versions of NetBSD 1.2 and higher support Arp for Arcnet. If you are using one of these newer NetBSD versions then please select "Arp" mapping in Miami.

1.29 Miami.guide/NODE_GUI_INTERFACE_MTU

MTU

(This option is available for serial devices only. The MTU value for SANA-II devices is set through SANA-II parameters .)

Maximum Transfer Unit, i.e. the size of the largest packet transferred at a time.

Recommended 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.

Also note: For PPP the MTU value is not critical, i.e. your connection will still work if the MTU value you selected is higher or lower than the optimum value. However for (C)SLIP you must make sure that your MTU value is not higher than the MTU value at your Internet provider.

1.30 Miami.guide/NODE_GUI_INTERFACE_STP

SANA-II parameters

(This option is available for SANA-II devices only.)

The gadget "SANA-II parameters" pops up a window with SANA-II link level settings for the device. These settings include

* The hardware address of the device, with an option to override it. (For bus/ring devices only.) Hardware addresses are specified as a sequence of bytes in hexadecimal notation, separated by ':', e.g.

'01:23:45:67:89:ab'.

* The link-level packet types for IP, Arp and RArp packets. (RArp is not available with Arcnet, and neither Arp nor RArp are available with point-to-point devices.)

* The MTU for the device.

* The number of IORequests used for IP and Arp packets. (Arp is not available with point-to-point devices.)

In most cases you should initialize all of these values to default values by clicking on "Query device" (only while Miami is offline). However you can manually override all values if necessary, e.g. if you are using a new hardware type for which Miami does not know the correct default settings.

1.31 Miami.guide/NODE_GUI_INTERFACE_MNIP

MNI parameters

(This option is available for MNI drivers only.)

The gadget "MNI parameters" pops up a window with MNI link level settings for the device. These settings include

* The hardware address of the device, with an option to override it. Hardware addresses are specified as a sequence of bytes in hexadecimal notation, separated by ':', e.g. '01:23:45:67:89:ab'.

* The link-level packet types for IP, Arp and RArp packets. (RArp is not available with Arcnet, and neither Arp nor RArp are available with point-to-point devices.)

* The MTU for the device.

In most cases you should initialize all of these values to default values by clicking on "Query device" (only while Miami is offline). However you can manually override all values if necessary, e.g. if you are using a new hardware type for which Miami does not know the correct default settings.

1.32 Miami.guide/NODE_GUI_INTERFACE_MNIINFO

MNI information

(This option is available for MNI drivers only.)

The gadget "MNI information" shows some information about the currently configured MNI driver, including version, copyright information, and a list of boards supported by the driver.

1.33 Miami.guide/NODE_GUI_INTERFACE_FINDB

Find boards

(This option is available for MNI drivers only.)

The gadget "Find boards" activates the currently configured MNI driver, and looks for Ethernet boards in your system that are supported by the driver. A window then pops up that displays all supported boards. Select the board you want to use and click on "OK". This automatically sets the unit number correctly.

1.34 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.

This function only allows you to prevent hangups at times of inactivity. Some users want to do the opposite: enforce hangups at times of inactivity, to save on telephone costs. The tool MiamiRemind allows you to do that.

1.35 Miami.guide/NODE_GUI_PPP

PPP
===

1.36 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.

Registered users who have installed MiamiSSL 1.2 or higher can enable 'Allow MS-CHAP'. This improves compatibility with some misconfigured Windows-NT PPP servers. If this option is disabled then Miami falls back to using PAP when the server requests MS-CHAP.

1.37 Miami.guide/NODE_GUI_PPP_CALLBACK

Callback

(This function is available in the registered version only.)

PPP supports callback ('dialback') according to the CBCP protocol. If your provider is configured for it, then you can negotiate with your provider to call you back in order to save on telephone costs.

Depending on the configuration at your provider you either need to choose 'CBCP fixed', in which case your provider calls you back to a predefined phone number, or 'CBCP variable', in which case your provider calls you back to the phone number you enter in the gadget below.

'Min delay' is the delay you ask the provider to wait before calling you back. This should be large enough to allow your modem to hang up the line and reinitialize itself.

'Max delay' is the maximum delay you want Miami to wait for a callback before giving up.

1.38 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.

1.39 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 with some GUI modules click on the popup gadgets to toggle each control character individually.

1.40 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 (by either setting it to "RAM" or to "file") 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 set up to survive a reboot (for the "RAM" setting) or from a file on harddisk (for the "file" setting). In most cases this allows you to reconnect to your provider after rebooting your Amiga.

Please note: If you use the "file" setting and your Amiga crashes (for whatever reason, e.g. caused by a run-away commodity or patch) while Miami is writing the reconnect-file to harddisk, then it is possible that your harddisk gets invalidated or damaged in some way, caused by some bugs and other shortcomings in the Amiga filesystem.

It is therefore safer to use "RAM", because then Miami does not have to create a harddisk file. However the "RAM" setting 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 the "RAM" setting does not cause a proper reconnect.

1.41 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.

1.42 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 then you might have to disable this option.

1.43 Miami.guide/NODE_GUI_PPP_TERMREQ

```
TermReq before Hangup
---------------------

   This option should normally be switched on. In this case Miami sends
an LCP-TermReq message to your provider when you want to hang up the
line. This usually has the effect that your provider hangs up the modem
first, causing your modem to hang up more quickly.

   However some PPP servers do not support LCP-TermReqs properly. If
you notice that hanging up the line takes very long then try disabling
this option and see if hangups are quicker this way.
```

1.44 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

                 Redial Delay
                                              The ‘Redial Delay’ gadget

                 Teach
                                                     The ‘Teach’ gadget

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

                 Capture
                                                   The ‘Capture’ gadgets
```

1.45 Miami.guide/NODE_GUI_DIALER_SCRIPT

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

The 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. if you press 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.

1.46 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.

1.47 Miami.guide/NODE_GUI_DIALER_MAX

Max Repeat

If no connection can be established with any of the listed phone numbers, 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.

1.48 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.
```

1.49 Miami.guide/NODE_GUI_DIALER_RDELAY

```
Redial Delay
------------

   This value specifies the delay between successive dial attempts (for
different phone numbers). Usually you want this value to be zero, i.e.
have Miami dial one number immediately after the first number was busy.

   However some European modems require minimum delays between
successive dial attempts. If you have one of these modems then you need
to set the "Redial Delay" to a value large enough for your modem.
```

1.50 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.
```

1.51 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.
```

1.52 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.
```

1.53 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. For each part of the database you see
a listview and a set of string gadgets to modify the current entry.

   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.

   In the registered version you can also sort sections of a database,
import/export from/to the Clipboard, and merge the database with ASCII
files.

   With the MUI user interface modules you can rearrange entries of the
database by dragging them off the side of the listview and then moving
them back into the listview at their intended position. Please see the
MUI documentation for more information on drag-sorting listviews.

   Each entry in the database can be individually enabled or disabled.
Enabled entries are indicated by a `>>' marker to the left of the
entry. You can enable or disable entries by double-clicking on them
(with most GUI modules), or by selecting an entry and then clicking on
`Enable' or `Disable'.

   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.
```

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
Arp	The ‘Arp’ part
Socks	The ‘Socks’ part
IP filter	The ‘IP filter’ part

1.54 Miami.guide/NODE_GUI_DATABASE_PROTOCOLS

```
Protocols
---------
```

List of all supported protocols (relative to IP), consisting of a protocol name, a protocol ID, and an optional list of aliases. The table corresponds to the "etc/protocols" or "db/protocols" file in other protocol stacks.

This table hardly ever has to be changed. You should never remove one of the default entries from this table.

1.55 Miami.guide/NODE_GUI_DATABASE_SERVICES

```
Services
--------

   List of all supported services (TCP or UDP), consisting of a service
name, a service ID, a protocol name, and an optional list of aliases.
The table corresponds to the "etc/services" or "db/services" file in
other protocol stacks.

   Some application programs might require changes (usually additions)
to this list. However you should never remove one of the default
entries from this table.

   In particular: removing one entry from this table is not the proper
way of disabling its function in InetD. If you want to disable a server
in InetD then remove it from the "InetD" table, or disable it in the
"InetD" table, but do not remove it from the "services" table.
Otherwise you might get spurious errors from other applications later.
```

1.56 Miami.guide/NODE_GUI_DATABASE_HOSTS

```
Hosts
-----

   List of all host names (and corresponding IP addresses), consisting
of an IP address, a host name, and an optional list of aliases.  The
table corresponds to the "etc/hosts" or "db/hosts" file in other
protocol stacks.

   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. However you should only add
mappings for names that are under your personal control. Never add
mappings for hosts elsewhere on the Interent, because otherwise you
would be unable to contact those hosts when they are renumbered.
```

1.57 Miami.guide/NODE_GUI_DATABASE_NETWORKS

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

   List of all networks, consisting of a network name, a network ID, and
an optional list of aliases.  The table corresponds to the
"etc/networks" or "db/networks" file in other protocol stacks.
```

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

1.58 Miami.guide/NODE_GUI_DATABASE_DOMAINS

Domains

List of all local domains, specified by just the domain name. The table corresponds to the "etc/domains" or "db/domains" file in other protocol stacks.

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.

Note that abbreviating host names this way only works for names looked up through DNS, not for names looked up through the "Hosts" table. This means if you for instance add a domain "foo.edu", have a host "ex1.foo.edu" at 10.0.0.1 and want to be able to access that host by just typing "ex1", then you need to add an alias "ex1" for the host in the "Hosts" table as well (i.e. make the "Hosts" table entry "10.0.0.1 ex1.foo.edu ex1").

1.59 Miami.guide/NODE_GUI_DATABASE_DNSSERVERS

DNS servers

List of DNS servers, specified by just the IP address of the server.

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".

1.60 Miami.guide/NODE_GUI_DATABASE_INETD

```
InetD
-----

   List of daemons started by the built-in InetD consisting of a
service name (corresponding to an entry in the "services" table), a
socket type ("dgram" or "stream"), a wait mode ("wait", "nowait" or
"dos"), the owner (usually "root" for AmigaOS), the server's file name,
the server's process name, and a list of arguments to be sent to the
server.  The table corresponds to the "etc/inetd.conf" or
"db/inetd.conf" file in other protocol stacks.

   The InetD built-in to Miami supports many 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.

   For security reasons it is recommended that you disable the "echo",
"discard" and "chargen" services, because they can be abused for
denial-or-service attacks.
```

1.61 Miami.guide/NODE_GUI_DATABASE_USERS

```
                Users
-----

   List of users in the system, consisting of a user name, a password,
a user ID, a group ID (index into the "groups" table), a real name, a
home directory, and a command to be used to start a shell through
telnet.  The table corresponds to the "etc/passwd" or "db/passwd" file
in other protocol stacks.

   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 are not displayed in
the listview. The password column shows

`-'
     if no password is associated with a user, i.e. if login is possible
     without a password.

`*'
     if no login is possible to this account.

a centered `x'
     if a valid password is associated with this user.

   The procedure to enter passwords differs depending on the GUI module
```

you use. For MUI and some other modules click on the "Password" popup gadget to change the password. A string requester pops up then. For other module you need to enter the new password directly in the string gadget.

If you leave the string gadget empty then no password will be associated with the user (shown as '-'). If you enter just the single character '*' then logins will be inhibited (shown as '*'). In all other cases the text you type will be used as the password (shown as a centered 'x').

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
.

1.62 Miami.guide/NODE_GUI_DATABASE_GROUPS

Groups

List of groups in the system, consisting of a group name, a group ID and an optional user list. The table corresponds to the "etc/group" or "db/group" file in other protocol stacks.

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.

1.63 Miami.guide/NODE_GUI_DATABASE_ARP

Arp

List of manual Arp entries in the system, consisting of an IP address and a hardware address. The hardware address has to be specified in the usual colon-hex notation (e.g. '01:23:45'). The table corresponds to the "etc/ethers" or "db/ethers" file in other protocol stacks.

Arp is only used with bus/ring-type SANA-II devices, and you only need to add Arp entries manually if one of the other machines on the local network does not support Arp.

1.64 Miami.guide/NODE_GUI_DATABASE_SOCKS

Socks

List of SOCKS configuration entries in the system, consisting of a protocol type, a command, a list of hosts, a list of ports, and a list of proxies. The table defines which proxy (SOCKS) server, if any, is supposed to be contacted, as a function of the host and port to connect to.

Most users do not have to make any changes in this table. If you do not use SOCKS at all then just ignore this table. If you do use SOCKS then in most cases it is sufficient to leave this table empty, and only configure a SOCKS server in
Socks
. You only need to make changes in
this table if you want Miami to contact different SOCKS servers for different hosts or ports, or if you have a complicated local network (with multiple subnets) inside of the SOCKS firewall.

Each entry in this table defines a filter for a connection or bind attempt, and a list of proxy servers that are supposed to be contacted for connections matched by the filter. For each connection or bind attempt the table is scanned from beginning to end, and the first match is used, i.e. the order of entries in this table is significant. The format of each entry is as follows:

Type
: This defines the type of connection to be used, if this filter entry matches. Valid values are ‘socks4’ for a SOCKS V4 connection, ‘socks5’ for a SOCKS V5 connection and ‘noproxy’ for a direct connection, without SOCKS.

Command
: This field is part of the filter, and can be a comma-separated list of letters, with no white space in between. Each letter indicates one type of request: ‘c’: connect. ‘b’: bind. ‘u’: UDP. ‘p’: ping. ‘t’: traceroute. ‘-’: any request.

Hosts
: This field is part of the filter, and can be a host definition, as follows: ‘hostip/mask’: matches a range of target hosts by IP address and netmask, e.g. ‘1.2.3.4/255.255.0.0’. ‘-’: matches all hosts. ‘n1’: equivalent to ‘n1.0.0.0/255.0.0.0’. ‘n1.n2’: equivalent to ‘n1.n2.0.0/255.255.0.0’. ‘n1.n2.n3’: equivalent to ‘n1.n2.n3.0/255.255.255.0’. ‘.domain.name’: matches all hosts ending in ‘.domain.name’. ‘a.host.name’: matches exactly the host ‘a.host.name’.

Ports
: This field is part of the filter, and can be a port definition, as follows: ‘-’: matches all ports. ‘name’: matches the named service, e.g. ‘ftp’. ‘number’: matches the given port number, e.g. ‘80’. ‘[100,1000]’: matches ports 100 through 1000. ‘(100,1000)’: matches ports 101-999. ‘(100,1000]’: matches ports

101-1000.

Proxies
This defines which proxy servers to contact for requests that match this filter. It can be a comma-separated list of server entries. Each server entry has to be specified by hostname or IP address, optionally followed by a colon and the port number of the proxy server.

This table is only functional if 'SOCKS' is enabled in
Socks
. For
requests not matched by this table the default behavior is to contact the SOCKS server/port defined in
Socks
using SOCKS5.

1.65 Miami.guide/NODE_GUI_DATABASE_IPFILTER

IP filter

(This function is only available in the registered version.)

This table allows you to filter out some of the IP packets arriving at the local interface, or to create system log entries if some specific packets arrive. This allows you to implement a very rudimentary firewall, or to be notified when someone tries to break into your machine.

The table consists of a sequence of rules. Each packet that arrives is checked against each of the rules, from top to bottom. The first rule that applies to the packet dictates whether the packet is filtered out, and whether a log entry for this packet is generated for this packet. Rules further below in the table are not checked.

Each entry in the table consists of the following parts:

* A protocol, i.e. 'tcp', 'udp' or '*' meaning 'any protocol'.

* A service, i.e. a name that appears in the 'services' table, '*' meaning 'any port' or '$' meaning 'any service port', i.e. any port not in the range from 1024-5000). It is also possible to specify a range of services here, using the '/' as the delimiter between the first and last service, e.g. '1/80' is the range from port 1 to port 80.

* An IP address, refering to the source IP address of the packet.

* A netmask, defining the range of IP addresses

* Two parameters that define the action: you can allow or disallow access ('y' or 'n'), and create or inhibit a log entry ('y' or

'n').

Note that log entries are only created for 'tcp' services, not for 'udp' services.

Here is an example of a useful start configuration for the IP filter:

```
* * 127.0.0.1 (empty mask) y n
tcp auth *.*.*.* (empty mask) y n
* $ *.*.*.* (empty mask) y y
```

What this does is:

The first line ensures that any packet sent locally (i.e. from your Amiga to yourself) is allowed without logging.

The second line also allows incoming 'auth' requests without logging. This is useful because 'auth' ('identd') requests are issued by so many httpd, ftpd and ircd servers that you probably do not want to be bothered with a log entry for each request.

The third line allows all remaining external requests, but generates a log entry, telling you that someone is trying to access your machine. It is important that the service is specified as '$', not '*'. That's because ftp uses reverse-connects (from the server to the client) during upload and download. If you specified the service as '*' then you would also get a log entry each time you download or upload a file from/to an ftp server.

All remaining packets (i.e. packets from the outside sent to a port between 1024 and 5000) use the implied default rule, which is to allow the packet and not to generate a log entry.

1.66 Miami.guide/NODE_GUI_TCPIP

```
                TCP/IP
======
                                 The ‘Verify DNS servers’ gadget

                Fake IP
                                            The ‘Fake IP’ gadget

                T/TCP
                                              The ‘T/TCP’ gadget

                Auto-add domain
                                    The ‘Auto-add domain’ gadget

                Down when offline
                                  The ‘Down when offline’ gadget

                Ping flood protection
                              The ‘Ping flood protection’ gadget

                Get time
                                           The ‘Get time’ gadgets
```

1.67 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 users that is not listed in the DNS. In
this case switch "dynamic" off and enter your host name manually.
```

1.68 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.
```

1.69 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.
```

1.70 Miami.guide/NODE_GUI_TCPIP_BOOTP

```
Use DHCP
--------

   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 "DHCP" (or its predecessor "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 "DHCP" on, and then later try again with
"DHCP" switched off, and see if this still works.

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

1.71 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 or 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.
```

1.72 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.
```

1.73 Miami.guide/NODE_GUI_TCPIP_TTCP

```
T/TCP
-----

   (This option is available in the registered version only.)

   T/TCP (TCP for Transactions) is an extension to TCP that can
significantly increase the speed of some types of applications, in
particular of web browsers, if the browser and the server both support
T/TCP.

   Registered users should usually enable this option to make use of the
speed advantage. However a few PPP servers have problems with the
extended TCP packets generated by T/TCP, so if Miami stops working
after enabling T/TCP you might have to disable this option - or switch
providers.
```

1.74 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.
```

1.75 Miami.guide/NODE_GUI_TCPIP_DOWN

```
Down when offline
-----------------

   (This option is available in the registered version only.)

   In the unregistered version Miami always disconnects all active TCP
sessions when the interface goes offline. In the registered version
Miami keeps TCP sessions alive in such a situation. This has the
advantage that you might be able to reconnect quickly and to continue
to use your TCP session.

   The drawback of keeping TCP sessions alive when the interface goes
offline is that applications cannot detect whether Miami is online or
offline, i.e. their connection attempts would just time out when Miami
is offline, but not generate any other error message.

   If you don’t like this behavior and would prefer Miami to generate
proper errors when the interface is offline then enable this option.
```

1.76 Miami.guide/NODE_GUI_TCPIP_PING

```
Ping flood protection
---------------------

   (This option is available in the registered version only.)

   Miami has a simple heuristic to reduce the effects of
denial-of-service attacks resulting from ping flooding:

   If this option is enabled and a user tries to ping-flood your
machine (either by sending very large pings or by sending pings very
quickly), Miami generates a log entry informing you about the attempt,
and stops generating ping responses to that user for a while, until the
user has stopped flooding you for some time.

   Note that there is no way for you to prevent the user from flooding
you, i.e. to stop him from wasting your modem bandwidth.  All Miami can
do in response to ping flooding is to stop sending responses and to
tell you about it (so you can reconnect to a different modem port). It
is not possible for Miami to prevent that user from wasting your modem
bandwidth. That would only be possible by installing a filter at your
Internet provider.
```

1.77 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. Please see
Time zone information
for
more details.

Note: Do not use this function if your Amiga has a battery-backed clock, because in this case it is possible that setting the time will cause your Amiga’s time to be changed backwards. This can confuse programs which use GetSysTime() for calculations, and can cause crashes and other problems.

1.78 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 or Shell 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

Shell
     Start an AmigaDOS shell script

Hide GUI
     iconify the Miami window

Kill GUI
     iconify the Miami window and unload the GUI module

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 options "ARexx" and "Shell" are not available, and "auto-online" is not available in response to a "passive offline" event.

The gadget "Console name" allows you define the input/output stream that ARexx and Shell scripts use. It should be something like "CON:1/1/400/100/title".

1.79 Miami.guide/NODE_GUI_MODEM

```
                Modem
=====
```

Init String	The ‘Init String’ gadget
Exit String	The ‘Exit String’ gadget
Dial prefix	The ‘Dial prefix’ gadget

1.80 Miami.guide/NODE_GUI_MODEM_INIT

Init String

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

1.81 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.

1.82 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".

1.83 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".

1.84 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.
```

1.85 Miami.guide/NODE_GUI_LOGGING

```
                Logging
=======



                 Console
                                    The ‘Console’ gadget

                 File
                                       The ‘File’ gadget

                 Use syslog.library
                         The ‘Use syslog.library’ gadget

                 Phone log
                                  The ‘Phone log’ gadgets

                 PPP log
                                    The ‘PPP log’ gadgets
```

1.86 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.
```

1.87 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.
```

1.88 Miami.guide/NODE_GUI_LOGGING_SYSLOG

```
Use syslog.library
------------------

   If you enable this gadget then Miami tries to access syslog.library
for its system log. syslog.library is part of the SysLog package by
Petri Nordlund.
```

1.89 Miami.guide/NODE_GUI_LOGGING_PHONE

```
Phone log
---------

   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.

1.90 Miami.guide/NODE_GUI_LOGGING_PPP

PPP log

(This option is available in the registered version only.)

The PPP log gadget allows you to specify a file name where Miami logs the connection establishment phase of PPP. Data is logged in human-readable form, i.e. not as a hex dump. Only the connection establishment phase is logged, i.e. until the LCP and IPCP state machines have reached the ‘Open’ state. After that logging stops.

The primary purpose of the PPP log is to help track down compatibility problems at the PPP level, and to help optimize PPP options for a particular PPP server.

1.91 Miami.guide/NODE_GUI_WINDOWS

```
                Windows
=======



                 Quit requester
                                                 The ‘Quit requester’ gadgets

                 Offline requester
                                              The ‘Offline requester’ gadget

                 Error requester
                                                The ‘Error requester’ gadget

                 Dialer
                                                        The ‘Dialer’ gadgets
```

1.92 Miami.guide/NODE_GUI_WINDOWS_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.

1.93 Miami.guide/NODE_GUI_WINDOWS_REQOFFLINE

Offline requester

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

1.94 Miami.guide/NODE_GUI_WINDOWS_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.

1.95 Miami.guide/NODE_GUI_WINDOWS_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.

The "Activate windows" gadget tells Miami whether you want Miami to activate dial windows and error requesters when they pop up.

1.96 Miami.guide/NODE_GUI_GUI

```
                GUI
===
```

This page defines the user interface settings for Miami, i.e. hotkeys, iconification, icons and the user interface engine to be used.

Important: Always specify user interface settings on this page, not in any other preferences program. Even if you use MUI then do not use the iconify and hotkey functions in MUI preferences for Miami. These functions do not work with Miami, because Miami handles iconification by itself, without MUI.

Hotkey	The ‘Hotkey’ gadget
Show icon	The ‘Show icon’ gadget
Show menu	The ‘Show menu’ gadget
No GUI on startup	The ‘No GUI on startup’ gadget
Online icon	The ‘Online icon’ gadget
Offline icon	The ‘Offline icon’ gadget
GUI engine	The ‘GUI engine’ gadget
Switch	The ‘Switch’ gadget

1.97 Miami.guide/NODE_GUI_GUI_HOTKEY

```
Hotkey
------
```

This string gadget specifies the Commodities hot key to iconify or deiconify the Miami user interface. Standard Commodities syntax is used to define the hot key, e.g. ‘ctrl alt m’ defines the hot key to be the key ‘m’, pressed together with the ‘ctrl’ key and either ‘alt’ key. ‘ctrl alt m’ is also the default.

1.98 Miami.guide/NODE_GUI_GUI_SHOWICON

Show icon

If this gadget is checked then an AppIcon is displayed on the Workbench screen when Miami is iconified.

1.99 Miami.guide/NODE_GUI_GUI_SHOWMENU

Show menu

If this gadget is checked then a ‘Miami’ entry in the Workbench ‘Tools’ menu is created when Miami is iconified.

1.100 Miami.guide/NODE_GUI_GUI_ONSTARTUP

No GUI on startup

If this gadget is checked then Miami does not load the user interface module on startup, and does not open its window. This feature is most useful if you combine it with ‘automatic online on startup’. See Events for more information on this.

1.101 Miami.guide/NODE_GUI_GUI_ONLINEICON

Online icon

This gadget lets you specify an icon (‘.info’ file) that Miami uses as the AppIcon whenever Miami is online. The default (empty gadget) is to use a built-in image.

1.102 Miami.guide/NODE_GUI_GUI_OFFLINEICON

```
Offline icon
------------

   This gadget lets you specify an icon (`.info' file) that Miami uses
as the AppIcon whenever Miami is offline. The default (empty gadget) is
to use a built-in image.
```

1.103 Miami.guide/NODE_GUI_GUI_GUI

```
                GUI engine
----------

   This gadget lets you choose one of several installed GUI engines.
Your choice is remembered by Miami and stored in the settings file (if
you save settings afterwards), but Miami does not immediately switch to
the new GUI engine. To do that click on
                Switch
                .
```

1.104 Miami.guide/NODE_GUI_GUI_SWITCH

```
Switch
------

   Clicking on this gadget causes Miami to switch to the selected GUI
engine. (What really happens is: Miami iconifies, removes the current
GUI module, loads the new GUI module, and then deiconifies with the new
GUI module.)
```

1.105 Miami.guide/NODE_GUI_SOCKS

```
                Socks
=====

   This page lets you configure SOCKS client support in Miami. If you
have never heard about SOCKS then you probably don't need it. SOCKS is
a proxy system to allow sites within a firewall to make connections to
machines outside of the firewall.

   Miami's SOCKS implementation allows Amiga TCP/IP clients to connect
"through" firewalls transparently, without special support in the
clients. If your network provider uses a SOCKS firewall then ask them
for the IP address of the SOCKS server, and for the SOCKS username and
```

password (if the SOCKS server is password-protected), and configure
Miami for SOCKS on this page.

The settings on this page are default settings for your setup. You
can configure the SOCKS settings in more detail in
Database/Socks
.

Enable SOCKS	The 'Enable SOCKS' gadget
Default SOCKS Server	The 'Default SOCKS server' gadgets
Maximum Syslog level	The 'Maximum Syslog level' gadget
Authentication	The 'Authentication' gadgets

1.106 Miami.guide/NODE_GUI_SOCKS_ENABLE

Enable SOCKS

If this gadget is enabled then Miami uses SOCKS to connect to any
machine not directly reachable through any interface. You also need to
configure the SOCKS server IP address, port and, with some SOCKS
servers, authentication.

1.107 Miami.guide/NODE_GUI_SOCKS_SERVER

Default SOCKS server

These gadgets define the IP address and port number of the default
SOCKS server in your network. The port number for SOCKS is usually 1080.

1.108 Miami.guide/NODE_GUI_SOCKS_MAXLOG

Maximum Syslog level

This gadget defines how many diagnostic messages you want to receive from the SOCKS wrapper. You should usually keep this gadget at "none" or "error". Higher settings are useful to get additional diagnostic messages during debugging.

1.109 Miami.guide/NODE_GUI_SOCKS_AUTH

Authentication

These gadgets specify the authentication data sent to the SOCKS server. The following authentication methods are possible:

none
: No authentication data is sent. This only works with SOCKS servers that do not require authentication.

same as in dialer
: Miami sends the username/password combination you defined in the dialer to the SOCKS server.

username/password
: Miami sends the username/password combination specified in the gadgets below to the SOCKS server.

1.110 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 or a set of buttons on the left side of the Miami window, that is used to select one of the configuration pages.

1.111 Miami.guide/NODE_MNI

MNI Ethernet drivers
****

MNI drivers are a new way of accessing your Ethernet board. Compared to traditional SANA-II drivers MNI usually offers better performance (often MUCH better performance), additional features, e.g. support for promiscuous mode in MiamiTCPDump, and easier configuration. The compatibility with some types of hubs and cable modems is also higher in some situation than with SANA-II.

The drawback of using MNI is that MNI is not supported by other networking stacks yet (e.g. Envoy), so you cannot run other stacks in parallel with Miami while using MNI. This will change in the future though.

To use your Ethernet board with MNI, set the interface type in Miami to "MNI Ethernet", and enter the name of the MNI driver for your Ethernet board (see the list below). Then click on "Find boards" to make sure your board is supported, select the board and click on "OK". This sets the unit number correctly. Now click on "MNI parameters" and "Query device", then "OK" to configure link-level settings. After that configure the remaining settings (e.g. IP addresses). For some MNI drivers you may also have to enter options in "MNI Options". Please see the description of your MNI driver to find if this is necessary for your driver.

Here is a list of all currently supported Amiga Ethernet boards, along with pointers to the corresponding MNI driver.

ASDG LanRover EB920
ASDG LanRover EB920: z2-dp8390.mni

Ariadne
Ariadne: z2-am7990.mni

Ariadne-II
Ariadne-II: z2-dp8390.mni

CEI/Ameristar A2065
CEI/Ameristar A2065: z2-am7990.mni

CEI/Ameristar A4066
CEI/Ameristar A4066: z2-smc91c90.mni

Commodore A2065
Commodore A2065: z2-am7990.mni

ConneXion
ConneXion: z2-am7990.mni

GG2-Bus+, NE2000
GG2-Bus+, NE2000: gg2-dp8390.mni

Hydra AmigaNet Z2
Hydra AmigaNet Z2: z2-dp8390.mni

QuickNet QN2000
QuickNet QN2000: z2-mb86950.mni

Here is a list of all currently available MNI drivers, along with configuration information.

```
                 gg2-dp8390.mni
                                      gg2-dp8390.mni

                 z2-am7990.mni
                                       z2-am7990.mni

                 z2-dp8390.mni
                                       z2-dp8390.mni

                 z2-mb86950.mni
                                      z2-mb86950.mni

                 z2-smc91c90.mni
                                     z2-smc91c90.mni
```

1.112 Miami.guide/NODE_MNI_GGTDPETNZ

```
gg2-dp8390.mni
==============

   Driver for NE2000-compatible ISA boards (based on the DP8390 chip or
clones), in Amiga ISA slots bridged via the GG2-Bus+ board.

   The driver should support all NE2000-compatible (NOT
NE1000-compatible) boards. This includes boards configured via jumpers,
boards configured via installation disk (with the configuration saved
in Flash-ROM), and jumperless ISA-PnP boards without a fixed
configuration.

   The MNI driver auto-detects the board (IO address and IRQ).  Manual
configuration is not supported at this time. If no board is found in
the supported IO and IRQ range then the driver performs a PnP search
for NE2000-compatible boards, and configures and activates any found
board for the time period the driver is active.

   Currently only one single NE2000 board in ISA slots will be
recognized.

   Supported MNI options:

   * One of "FT=0", "FT=1", "FT=2", "FT=3" selects the FIFO threshold
     for local DMA. The default is "FT=2". You should usually not need
     to change this value.

   * One of "WAIT=1", "WAIT=0" enables/disables support for wait states
     on the GG2 board. Default is "WAIT=1". "WAIT=0" may slightly
     reduce CPU load with boards that are fast enough.
```

1.113 Miami.guide/NODE_MNI_ZTAMSNNZ

```
z2-am7990.mni
=============
```

Driver for the AM7990 (LANCE), AM79C90 (C-LANCE) and AM79C960 (PC-net) chips, on the Zorro bus. Supported boards at this time:

* Ariadne (NOT Ariadne-II)

* CEI/Ameristar A2065

* Commodore A2065

* ConneXion in A2065-compatibility mode

Important information for users of ConneXion boards: Make sure the jumper on your board is configured for A2065 compatibility. The driver does not support the ConneXion native mode at this time.

Important information for users of A2065 boards and ConneXion boards: When switching from SANA-II to MNI drivers you need to first change the configuration to MNI in Miami, save the settings, and then reboot your computer before going online again. That is necessary because A2065 SANA-II drivers can only be removed from memory by rebooting the computer.

Important information for users of Ariadne boards: Your Amiga may already have an Ariadne driver in "SYS:Expansion". If so then you need to remove that driver and reboot your Amiga before going online with the new MNI driver. Otherwise your Amiga may "hang".

Supported MNI options:

* Ariadne only: MEDIA=AUTO (Default. Enables automatic media detection)

* Ariadne only: MEDIA=10BASE2 (Selects 10-Base-2, i.e. Coax, BNC, Cheapernet cabling)

* Ariadne only: MEDIA=10BASET (Selects 10-Base-T, i.e. RJ45, UTP cabling)

1.114 Miami.guide/NODE_MNI_ZTDPETNZ

```
z2-dp8390.mni
=============
```

Driver for the DP8390 chip (and its numerous clones, e.g. integrated chips used for NE2000-compatible boards), on the Zorro bus. Supported boards at this time:

* ASDG LanRover EB920

* Hydra AmigaNet

* Ariadne-II (NOT Ariadne)

Please note that at the time this documentation was written Ariadne-II support was disabled in the driver included in Miami 3.2, because no board was available for testing. Support for the Ariadne-II is implemented though and will be enabled as soon as a board is available for testing.

Important information for ASDG LanRover EB920 owners: The board has a jumper that selects the interrupt (2 or 6). You MUST use an MNI option that corresponds to the jumper setting on your board. Otherwise your Amiga will crash. Also, some EB920 boards do not have an on-board MAC address ROM. If your board is among them then the MAC address returned by "query device" in "MNI parameters" is 00:00:00:00:00:00. In that case you need to enter a dummy address and select "Override". Please consult the documentation of your EB920 board for more information on this.

Important information for Hydra AmigaNet owners: With a reasonably fast CPU (040 or higher) this driver typically achieves very high throughput (> 800 kB/s) on local networks. If you get poor performance (500 kB/s or less) on a local network then your Hydra board is most likely defective. Unfortunately quite a lot of Hydra boards seem to have this problem. Such poor performance is NOT the result of a bug in the driver.

Supported MNI options:

* One of "FT=0", "FT=1", "FT=2", "FT=3" selects the FIFO threshold for local DMA. The default is "FT=2". You should usually not need to change this value.

* ASDG LanRover EB920 only: One of "INT=2" or "INT=6". This option MUST be set to correspond with the jumper setting on the board.

1.115 Miami.guide/NODE_MNI_ZTMBESNFZ

z2-mb86950.mni
==============

Driver for the Fujitsu MB86950 Ethernet chip, on the Zorro bus. Supported boards at this time:

* QuickNet QN2000

Important information for QuickNet QN2000 owners: The board has a switch on the back, which changes the product ID between 1 and 2. Both product IDs are supported, but product ID 2 tends to work better. It is therefore recommended that you configure your board for product ID 2, if possible. To do this first check which product ID you have (click on "Find Boards" in Miami). If the product ID is 1 then toggle the switch, reboot your Amiga, and try again. Not all boards can be set to product ID 2. Some only support product ID 1, in all switch settings.

Supported MNI options: none.

1.116 Miami.guide/NODE_MNI_ZTSMCNOCNZ

z2-smc91c90.mni
===============

Driver for the SMC91C90 chip (and its successors, e.g. the SMSC LAN91C94), on the Zorro bus. Supported boards at this time:

* CEI/Ameristar A4066

Supported MNI options: none.

1.117 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.

ASKPASSWORD
: Pop up a requester asking the user for the password.

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.

SENDPAD "text",padding
     Send <text> to the modem, padded with spaces up to a total length
     of <padding>. Example: ‘SENDPAD "abc",5’ would send "abc  ".

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.

WAITCONNECT
     Wait for a CONNECT message and the following text (usually the
     connection speed) to be received from the modem. This is identical
     to ‘WAIT "CONNECT"’, except that Miami copies anything following
     the ‘CONNECT’ message on the same line to an internal buffer, and
     later displays it in the status area. With many modems this allows
     you to see the speed at which your modem connected.

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.
```

1.118 Miami.guide/NODE_AREXX

ARexx Interface

The name of the Miami ARexx port is "MIAMI.1". At the moment Miami supports all of the standard ARexx commands for MUI applications ("QUIT", "HIDE", "DEACTIVATE", "SHOW", "ACTIVATE", "INFO", "HELP") plus the following additional commands:

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.

GETCONNECT
: Returns the connection string that followed the ‘CONNECT’ message from the modem. Usually this string contains an indication of the connection speed.

GETCONNECTTIME
: Returns the number of seconds since Miami received the ‘CONNECT’ message from the modem.

GETONLINETIME
: Returns the number of seconds Miami has been online in the ‘result’ variable.

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

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

KILLGUI
: Iconifies Miami’s windows and unloads the current GUI module.

LOADSETTINGS file/a
: Loads the specified settings file.

LOCKGUI
: Locks the user interface, i.e. displays a busy pointer. Calls to this function nest.

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

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

QUITFORCE
: Using the "QUIT" command from an ARexx script is the safest way to quit Miami, because Miami only attempts to go offline and quit if no other ARexx scripts are still running, to avoid deadlocks. The disadvantage of this is that there may be timing problems if your

ARexx control is complex, involves multiple ARexx scripts (in particular scripts for earlier events), and one or more scripts are still running when the "QUIT" command is issued: Miami would then refuse to quit, even though it might be safe to wait and quit later. In that case try the "QUITFORCE" command: it forces Miami to wait until all ARexx scripts have completed, and then quit. Warning: this command will lock up Miami if one of the pending ARexx scripts never returns, e.g. because of an inifite loop or a recursive call, so it is potentially dangerous if your ARexx scripts are buggy.

UNLOCKGUI
: Unlocks the user interface, i.e. removes the busy pointer and replaces it with the normal mouse pointer if no more instances of LOCKGUI are pending.

1.119 Miami.guide/NODE_ENVVARS

Environment variables
****

Time zone information
Time zone information

Users usually do not need to set any environment variables in order to use Miami. Nevertheless here is a list of all variables Miami uses, in case you want to make manual changes:

DOMAIN, DOMAINNAME
: These variables are set automatically by Miami, whenever Miami goes online. They are set to your current domain (i.e. to the part of your host name which follows the first ".").

HOME
: This variable is set automatically by Miami, whenever Miami goes online. It is set to the home directory configured in Database/Users, for the user you selected on the TCP/IP page.

HOST, HOSTNAME
: These variables are set automatically by Miami, whenever Miami goes online. They are set to your configured host name (for static host names), or to the host name corresponding to your IP address, found by Miami through reverse DNS lookup. If no host name was found then these variables are set to your IP address.

MagicWB
: If no user interface is specified (by the user, in the settings file, or in "ENV:MIAMI/GUI") then Miami falls back to using either "MUI" or "MUIMWB" as the default GUI. "MUIMWB" is used if the "MagicWB" variable exists, indicating that MagicWB has been installed.

```
REALNAME
     This variable is set automatically by Miami, whenever Miami goes
     online. It is set to the real name configured on the TCP/IP page.

SOCKETCONFIG
     This variable is set automatically by Miami, whenever Miami goes
     online. It is required by the freeware "socket.library" emulation
     library (for I-Net-225-compatible software), and is set in a way
     that allows that library to work properly.

TZ
     This variable is read by Miami to find your current time zone.  It
     should be set correctly before installing Miami. Please see

                Time zone information
                 for more on this.

USERNAME
     This variable is set automatically by Miami, whenever Miami goes
     online. It is set to the user name configured on the TCP/IP page.

MIAMI/GUI
     This variable should contain the name of your default GUI engine
     (e.g. ‘MUI’, ‘MUIMWB’ or ‘GTLayout’). It is set automatically
     during installation.

MIAMI/SSLLIB
     This variable is only needed when you use MiamiSSL, and is set
     automatically during the installation of MiamiSSL. It should
     contain the name of your SSL encryption library, i.e. either
     ‘Miami:Libs/miamisslintl.library’ or
     ‘Miami:Libs/miamisslusa.library’.
```

1.120 Miami.guide/NODE_ENVVARS_TZ

```
Time zone information
*********************

   The environment variable TZ has to be set as follows:

   Outside of daylight savings time (i.e. during winter):

   EST5

   where EST is the name of your time zone, and 5 is the *negative*
time difference to UTC (i.e. if you are 2 hours east of UTC then the
value has to be -2, not 2). In Europe, Asia and Australia that number
is usually 0 or negative, in America it is positive.  See below for
examples.

   During daylight savings time (i.e. during summer):
```

```
   EST4EDT

   EST, 4: meaning the same as above. EDT is the name of your time zone
during daylight savings time. Also make sure that you adjust the number
(4 in the example) by one hour as required by your local daylight
savings time conventions.

   The name of the time zone does not really matter in either case. It
is important that the number is right, though, and that the number
starts *exactly* at the fourth character position.

   It is NOT correct to put some RFC-compliant time string into ENV:TZ,
i.e. something like "EST", "EST (-0500)" or "-0500" will NOT work.

   Some examples:

                    winter   summer

     US west coast     PST8     PST7PDT
     US east coast     EST5     EST4EDT
     Britain           WET0     WET-1WEDT
     most of western
     Europe            MET-1    MET-2MEDT
```

1.121 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:
```

1.122 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. A (r) indicates that this feature is only available in

```
the registered version, and ignored in the unregistered version.
"(m)", "(i)" and "(r)" 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
```

```
CALLBACKTYPE= (r)
     NONE / CBCPFIXED / CBCPVARIABLE

CALLBACKPHONE= (i) (r)
     phone_number

CALLBACKMINDELAY= (r)
     integer

CALLBACKMAXDELAY= (r)
     integer

ACCM=
     000a0000

VJC=
     Y / N

QUICKRECONNECT=
     Y / N

TERMREQ=
     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

DIALREDIALDELAY=
     redialdelay

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

TTCP= (r)
     Y / N

DBHOSTS= (m)
     host_database

DBNETWORKS= (m)
     network_database

DBDOMAINS= (m)
     domain_database

DBDNSSERVERS= (m)
     dns_servers_database
```

1.123 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.

1.124 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.

You can add entries to any of Miami's database tables this way. However for security reasons only those tables which are commonly used by clients ('inetd' and 'services') are directly changed by Miami. If applications try to change any other table (e.g. the sensitive 'users' table), then Miami shows a requester, asking the user for confirmation, after receiving the "CHANGEDB" command.

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.

1.125 Miami.guide/NODE_UTILITY

```
                Utility Programs
****************
```

```
                 MiamiRoute
                                              MiamiRoute

                 MiamiSysCtl
                                             MiamiSysCtl

                 MiamiTCPDump
                                            MiamiTCPDump

                 MiamiTraceRoute
                                         MiamiTraceRoute
```

1.126 Miami.guide/NODE_UTILITY_ARP

```
MiamiArp
========

   Address resolution display and control

   Usage:

MiamiArp hostname
     Display current Arp entry for <hostname>

MiamiArp [-n] -a
     Display all of the current Arp entries. If "-n" is specified then
     all entries are listed numerically instead of symbolically.

MiamiArp -d hostname
     Delete arp entry for <hostname>

MiamiArp -s hostname hw_addr [temp] [pub]
     Create an Arp entry for <hostname> with the hardware address
     <hw_addr>. The entry is permanent unless the word "temp" is given.
     If the word "pub" is given then this system will act as an Arp
     server for the specified host.

MiamiArp -f filename
     Read and execute commands from the file <filename>.
```

1.127 Miami.guide/NODE_UTILITY_FINGER

```
MiamiFinger
===========

   MiamiFinger displays information about the system users.

   Usage: MiamiFinger [-l] [user][@machinename]
```

Options are:

-l
: Show the long output format (for remote machines: send the "/W" modifier to the remote finger daemon).

If no machine name is specified then "localhost" is assumed.

If a user is specified then information about this user is displayed. Otherwise some default information for the fingerd connecting to is displayed. In many cases this is some general system information and/or a list of users currently logged on.

This implementation of MiamiFinger supports T/TCP for faster finger lookups.

1.128 Miami.guide/NODE_UTILITY_IFCONFIG

MiamiIfConfig
=============

Configure network interface parameters

Note: most of the options of MiamiIfConfig should not be used with Miami at this time, because Miami usually already sets all values correctly. Do not play around with this program. You should really know what you are doing before trying to change any interface options.

About the only useful options are "up" and "down" to temporarily mark the interface as unavailable. Note that this does not cause the modem to hang up. Other than that you should probably only use MiamiIfConfig to examine interface settings, not to change them.

Usage: MiamiIfConfig interface [alias | -alias] [af [address [dest_addr]] [up] [down] [netmask mask]] [metric n] [arp | -arp] [broadcast address] [link0 | -link0] [link1 | -link1] [link2 | -link2]

interface
: Currently either "lo0" or "mi0"

alias/-alias
: Consider the specified address an alias for the existing address, i.e. do not overwrite an existing address.

af
: Address family: only "inet" is supported at this time.

address
: A protocol-level address. For the address family "inet" this is an IP address in dot-notation (e.g. 123.45.67.89).

dest_addr
: The protocol-level destination address. This is only used for point-to-point devices.

```
up/down
     Mark the interface as up or down.

netmask
     Change the netmask for this interface.

metric
     Change the metric (priority) for this interface. This has no
     effect for a single-interface stack like Miami.

arp/-arp
     Enable/disable Arp for this interface. This option should not be
     used with Miami. Use the Miami GUI instead to choose the type of
     address resolution.

broadcast
     Set the broadcast address for this interface.

linkx/-linkx
     Set or reset link-level flags 0, 1 or 2. These flags are not
     currently used by Miami.
```

1.129 Miami.guide/NODE_UTILITY_MAPMBONE

```
MiamiMapMBone
=============

   Multicast connection mapper

   Usage: MiamiMapMBone [-d debug_level] [-f] [-g] [-r retry_count] [-t
timeout_count] [starting_router]

   MiamiMapMbone attempts to display all multicast routers that are
reachable from the specified multicast starting_router. If not
specified on the command line, the default multicast starting_router is
the localhost.

   The options have the following meaning:

-d debug_level
     Sets the debug level. When the debug level is greater than the
     default value of 0, addition debugging messages are printed.

-f
     Sets flooding option. Flooding allows the recursive search of
     neighboring multicast routers and is enable by default when
     starting_router is not used.

-g
     Sets graphing in GraphEd format.

-n
     Disables the DNS lookup for the multicast  routers names.
```

```
-r retry_count
     Sets the neighbor query retry limit. Default is 1 retry.

-t timeout_count
     Sets the number of seconds to wait for a neighbor query reply
     before retrying. Default timeout is 2 seconds.
```

1.130 Miami.guide/NODE_UTILITY_MRINFO

```
MiamiMRInfo
===========

   Displays configuration info from a multicast router.

   Usage: MiamiMRInfo [-d debug_level] [-r retry_count] [-t
timeout_count] [multicast_router]

   MiamiMRInfo attempts to display the configuration information from
the specified multicast router. If no router is specified then
localhost is used.

   The options have the following meaning:

-d debug_level
     Sets the debug level. When the debug level is greater than the
     default value of 0, addition debugging messages are printed.

-r retry_count
     Sets the neighbor query retry limit. Default is 3 retries.

-t timeout_count
     Sets the number of seconds to wait for a neighbor query reply
     before retrying. Default timeout is 4 seconds.
```

1.131 Miami.guide/NODE_UTILITY_MROUTED

```
MiamiMRouteD
============

   IP Multicast Routing Daemon

   Usage: MiamiMRouteD [-p] [-c config_file] [-d debug_level]

   MiamiMRouteD is a program you might have to run in the background
("run MiamiMRouteD") to receive or forward multicast feeds. Please see
below for a more detailed explanation.

   The options have the following meaning:
```

```
-p
     Start MiamiMRouteD in non-pruning mode. This option should only be
     used for testing.

-c config_file
     Specify which config file to use. The default config file is
     "Miami:MiamiMRouteD.config".

-d debug_level
     Specify the debug level. The default is 0 (no debug info).

   MiamiMRouteD is a very complex and powerful program that allows you
to receive and forward multicast feeds. It is configured through a
separate configuration file, the format of which is partially described
below. However since Miami supports only a single interface, only few
features of MiamiMRouteD can reasonably be used with Miami, only few
users probably need to use it at all.

   The two most common configurations are:

   * You are receiving your multicast feed direcly from a broadcast- or
     multicast-capable interface such as Ethernet or Arcnet. In this
     case DO NOT run MiamiMRouteD. Instead enable multicasting in
     Miami, on the "Interface" page.

   * You are receiving your multicast feed through an IP tunnel,
     possibly via PPP from your provider. In this case disable
     multicasting in Miami for your PPP/SLIP interface, configure
     MiamiMRouteD for a tunnel to your provider (see below), and run
     MiamiMRouteD after starting Miami.

   The configuration file for MiamiMRouteD is a standard ASCII text
file. Each line can contain one command. The only command of interest
at the moment is the "tunnel" command, which allows you to configure an
IP tunnel to send and receive multicasts to/from. The important part of
the syntax is:

     tunnel <local-addr> <remote-addr>

   For <local-addr> you can specify an IP address or an interface name
(for Miami always "mi0"). <remote-addr> is the IP address of the host
on the other side of the multicast tunnel, e.g.

     tunnel mi0 1.2.3.4

   establishes a multicast tunnel to the host 1.2.3.4.
```

1.132 Miami.guide/NODE_UTILITY_MTRACE

```
                MiamiMTrace
===========

   Print multicast path from a source to a receiver
   Usage: MiamiMTrace [-g gateway] [-i if_addr] [-l] [-M] [-m max_hops]
[-n] [-p] [-q nqueries] [-r resp_dest] [-s] [-S stat_int] [-t ttl] [-v]
[-w waittime] source [receiver] [group]

   MiamiMTrace is a utility very similar to MiamiTraceRoute, but for
multicast addresses, not unicast addresses. Please see
                MiamiTraceRoute
                for more information on TraceRoute. "group" specifies the  ↩
                   multicast IP
address to use.  "source" and "receiver" are unicast IP address
specifiying the start and end point in the multicast path to trace. If
"group" is not specified then 224.2.0.1 is used. If "receiver" is not
specified then localhost is assumed.

   The options have the following meaning:

-g gateway
     Send the trace query via unicast directly to the specified
     multicast router rather than multicasting the query.  This must be
     the last-hop router on the path from the intended source to the
     receiver.

-i if_addr
     Use the specified address as the local interface address (on a
     multi-homed host) for sending the trace query and as the default
     for the receiver and the response destination.

-l
     Loop indefinitely printing packet rate and loss statistics for the
     multicast path every 10 seconds (also see '-S stat_int').

-M
     Always send the response using multicast rather than attempting
     unicast first.

-m max_hops
     Set the maximum number of hops that will be traced from the
     receiver back toward the source.  The default is 32 hops (infinity
     for the DVMRP routing protocol).

-n
     Print hop addresses numerically rather than symbolically and
     numerically (saves a nameserver address-to-name lookup for each
     router found on the path).

-q nqueries
     Set the maximum number of query attempts for any hop.  The default
     is 3.

-p
     Listen passively for multicast responses from traces initiated by
     others.  This works best when run on a multicast router.

-r resp_dest
     Send the trace response to the specified host rather than to the
     host on which MiamiMTrace is being run, or to a multicast address
     other than the one registered for this purpose (224.0.1.32).
```

```
-s
     Print a short form output including only the multicast path and not
     the packet rate and loss statistics.

-S stat_int
     Change the interval between statistics gathering traces to the
     specified number of seconds (default 10 seconds).

-t ttl
     Set the ttl (time-to-live, or number of hops) for multicast trace
     queries and responses.  The default is 64, except for local
     queries to the "all routers" multicast group which use ttl 1.

-v
     Verbose mode; show hop times on the initial trace and statistics
     display.

-w waittime
     Set the time to wait for a trace response to the specified number
     of seconds.
```

1.133 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.

   * MiamiNetStat [-AaDnN] [-f address_family]

   * MiamiNetStat [-dimnNrs] [-f address_family]

   * MiamiNetStat [-dnN] [-] [-I interface]

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

   * MiamiNetStat [-s] [-g]

   * 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 information about virtual interfaces (for multicasting) and multicast routing statistics.

The sixth 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.

-D
: With the default display, show the total number of transfered bytes for each active TCP connection.

-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.

-g
: Display the virtual interface table and multicast routing table. Together with the option ‘-s’ this option displays multicast routing statistics. Both of these options are only meaningful when MiamiMRouteD is running.

-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.

-N
: Only show a network address symbolically if the symbolic name is available without a prior DNS lookup. Otherwise show the network address as a number. 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.

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

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

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 (currently unused).

2
: RTF_PROTO2 Protocol specific routing flag #2 (currently unused).

3
: RTF_PROTO3 Protocol specific routing flag #3 (meaning for TCP: route is timing out).

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).

P
: RTF_PRCLONING Clone routes for use by protocols.

R
: RTF_REJECT Host or net unreachable.

S
: RTF_STATIC Manually added.

U
: RTF_UP Route usable.

W
: RTF_WASCLONED Route was created by cloning another route.

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, or the current SANA-II interface.

1.134 Miami.guide/NODE_UTILITY_PING

```
MiamiPing
=========

   Send packets to network hosts and listen for their response.

   Usage: MiamiPing [-Rdfnqrv] [-c count] [-i wait] [-l preload] [-p
pattern] [-s packetsize] hostname

   Options:

-c count
     Stop after sending and receiving <count> packets.

-d
     Set the SO_DEBUG option on the socket being used.

-f
     Flood ping. Outputs packets as fast as they come back, or one
     hundred times per second, whichever is more. For every ping sent a
     period "." is printed, while for every ping received a backspace
     is printed. This provides a rapid display of how many packets are
     being dropped. Note: Abusing this option for denial-of-service
     attacks is illegal.

-i wait
     Wait <wait> seconds between sending each packet. The default is to
     wait for one second between each packet. This option is
     incompatible with "-f".

-l preload
     Sends <preload> packets as fast as possible before falling into
     the normal mode of behavior.

-n
     Numeric output only.

-p pattern
     You may specify up to 16 "pad" bytes to fill out the packet you
     send.  This is useful for diagnosing data-dependent problems in a
     network.  For example,"-p ff" will cause the sent packet to be
     filled with all ones.

-q
     Quiet output. Nothing is displayed except the summary lines at
     startup time and when finished.

-R
     Record route. Includes the RECORD_ROUTE option in ping packets and
     displays the route buffer on returned packets. Note that the IP
     header is only large enough for nine such routes. Many hosts
     ignore or discard this option.
```

```
-r
     Bypass the normal routing tables and send directly to a host on an
     attached network.  If the host is not on a directly-attached
     network, an error is returned.  This option can be used to ping a
     local host through an interface that has no route through it
     (e.g., after the interface was dropped by routed).

-s packetsize
     Specifies the number of data bytes to be sent.  The default is 56,
     which translates into 64 ICMP data bytes when combined with the 8
     bytes of ICMP header data.

-v
     Verbose output.  ICMP packets other than ping response packets
     that are received are listed.
```

1.135 Miami.guide/NODE_UTILITY_REMIND

MiamiRemind
===========

Some users consider the automatic warning and disconnect after 30/60 minutes in the Miami demo version a useful feature, to save telephone/ISP costs.

MiamiRemind is a tool that introduces this kind of functionality in the registered version of Miami, but in addition to the simple warning/disconnect it offers several other useful features:

* The number of warnings, plus the interval between subsequent warnings can be freely configured.

* It is possible to disconnect after a certain amount of time, to only display a finite number of warnings (without disconnecting) or to keep displaying warnings at regular intervals.

* In addition to fixed time intervals it is possible to show warnings after a certain amount of *inactivity* on the link. Both types of warnings (warnings after fixed amounts of time and warnings after inactivity) can be enabled at the same time.

* Using the inactivity timer directly with the "disconnect" option provides the functionality of a "disconnect on inactivity" option, something many users have requested for Miami in the past.

The term "inactivity" is difficult to define for a TCP/IP connection. The default definition used by MiamiRemind is "lack of TCP traffic". With this definition MiamiRemind requires extremely little overhead and memory.

For users who need more sophisticated definitions of "inactivity", MiamiRemind provides an expression parser and compiler identical to the one in MiamiTCPDump, i.e. you can e.g. use expressions like

```
   "(tcp[13] & 3 != 0) or udp"

   The above expression would consider all TCP SYN packets, all TCP FIN
packets, and all UDP packets "activity".  All other packets are not
considered.

   The expression parser/compiler requires miamibpf.library and
miamipcap.library, and thus introduces slightly higher memory and cpu
overhead than the hardcoded "TCP traffic" definition.

     Usage: MiamiRemind [-f fixed_timer_spec]
                        [-i inactivity_timer_spec]
                        [-p pcap_spec]

   Option "-f" defines the parameters for the fixed timer, i.e.  the
timer that starts when MiamiRemind is started, without considering
activity on the link. The default is to disable the fixed timer.

   Option "-i" defines the parameters for the inactivity timer.  This
timer is reset to zero whenever a packet is transmitted or received
that is considered "activity" on the link. The default is to disable
the inactivity timer.

   Option "-p" defines the inactivity expression, in MiamiPCap format
(see the example above). The expression should be enclosed in double
quotes ("). If this parameter is specified then MiamiRemind uses
miamipcap.library and miamibpf.library to parse, compile and evaluate
the expression. Otherwise the hardcoded definition "TCP traffic" is
used, and both libraries are not needed.

   "timer_spec" (for options "-f" and "-i") is a string that consists
of numbers representing time intervals (measured in minutes), separated
by commas (",").

   Each time interval in the string represents the delay between
subsequent events.

   "event" usually refers to a warning requester. However it is also
possible to prefix numbers with the letter "D", to indicate that
MiamiRemind should disconnect the line at the next event, or with the
letter "L", to indicate that MiamiRemind should loop, i.e. use the next
time interval repeatedly, to define a sequence of events.

   Examples:

MiamiRemind -f 30,D30
     This is identical in behavior to the demo version of Miami, i.e.
     show a warning after 30 minutes, and disconnect after another
     thirty minutes.

MiamiRemind -f 30,20,L10
     Display a warning after 30 minutes, then again after 20 minutes,
     and from then on every 10 minutes (loop). Never disconnect the
     line.

MiamiRemind -f 60,60 -i L10
```

```
     Display a warning after 60 minutes and another one after another
     60 minutes. After that disable the fixed timer.  Also show a
     warning whenever there have been multiples of 10 minutes of
     inactivity (lack of TCP traffic) on the link.

MiamiRemind -i D30
     Disconnect the link after 30 minutes of inactivity (lack of TCP
     traffic).

MiamiRemind -i D20 -p "tcp or udp"
     Disconnect the link after 20 minutes of inactivity.  "inactivity"
     refers to TCP or UDP traffic.
```

MiamiRemind automatically quits when the interface goes offline (regardless of the reason), when Miami tries to quit, when the program receives a Ctrl-C signal, or when both timers are disabled.

The easiest way to use MiamiRemind is to start it directly from Miami whenever Miami goes online, i.e. as "run >nil: Miami:MiamiRemind [options]" in a Shell script launched from Miami (configured in Events->Online).

1.136 Miami.guide/NODE_UTILITY_RESOLVE

```
MiamiResolve
============
```

Resolve a host name to an IP address or an IP address to a host name.

Usage:

```
MiamiResolve ip_address
     Resolve the ip address, and display the associated host name and
     all ip addresses.

MiamiResolve host_name
     Resolve the host name, and display the associated host name and
     all ip addresses.

MiamiResolve -s port_number
     Resolve the port number, and display all associated service names
     and the port number.

MiamiResolve -s service_name
     Resolve the service name, and display all associated service names
     and the port number.
```

1.137 Miami.guide/NODE_UTILITY_ROUTE

```
MiamiRoute
==========

   Manually manipulate the routing tables.

   Usage: MiamiRoute [-nqv] command modifiers args

   Options:

-n
     Bypasses attempts to print host and network names symbolically
     when reporting actions.  (The process of translating between
     symbolic names and numerical equivalents can be quite time
     consuming, and may require correct operation of the network; thus
     it may be expedient to forgo this, especially when attempting to
     repair networking operations),

-q
     Suppress all output.

-v
     (verbose) Print additional details.

   Commands:

add
     Add a route

flush
     Remove all routes. Be very careful when using this command.  It
     also removes some of Miami's standard routes. Unless you repair
     this manually afterwards you will have to restart Miami to resume
     normal operation.

delete
     Delete a specific route

change
     Change aspects of a route (such as its gateway).

get
     Lookup and display the route for a destination.

monitor
     Continuously report any changes to the routing information base,
     routing lookup misses, or suspected network partitionings. Note:
     without an implementation of "routed" this command is not very
     useful.

   The MiamiRoute command is usually not needed with a single-interface
protocol stack like Miami, and very complex and difficult to use.  For
a complete discussion please see the BSD docs for the "route" command.

   About the only useful application of the "MiamiRoute" command at the
moment is to examine routes to hosts, e.g. to find out about round trip
times or path MTU values. To do this use the syntax:
   MiamiRoute get hostname

   To examine the complete routing table use the command "MiamiNetStat
-r", not MiamiRoute.
```

1.138 Miami.guide/NODE_UTILITY_SYSCTL

```
MiamiSysCtl
===========

   MiamiSysCtl lets you examine and change some of Miami's internal
variables.

   Usage:

MiamiSysCtl [-n] variable
     Examine the value of a variable.

MiamiSysCtl [-n] -w variable=value
     Change the value of a variable.

MiamiSysCtl [-n] -a
     Show a list of all variables.

MiamiSysCtl [-n] -A
     Show a list of all variables, plus additional information for
     domains not accessible through MiamiSysCtl.

   Options:

-n
     Show numerical output only.

   Explanation of all variables:

net.inet.ip.forwarding/redirect
     These options have no meaning on a single-interface stack like
     Miami.

net.inet.ip.ttl
     Controls the default ttl (time-to-live) for packets Miami sends.
     Should be at the default of 64.

net.inet.ip.rtexpire/rtminexpire/rtmaxcache
     Controls the timing and size for route cloning. You should not
     change these values.

net.inet.ip.sourceroute
     Controls the behavior for packets that contain an IP source route.
     This option is only important for routers.

net.inet.ip.pathmtudisc
     Specifies whether Path MTU Discovery is enabled (1/0). The default
     is to enable it, but if you are connected through old, buggy
     routers and have problems with TCP traffic then try disabling this
     option.

net.inet.icmp.maskrepl
     Controls whether Miami sends the netmask in response to ICMP mask
     queries. If the netmask is configured correctly and you enable
     this option, then any other machine on the local network running
     MiamiInit or Miami will be able to automatically find the correct
     netmask from ICMP.

net.inet.tcp.rfc1323
     Enables RFC1323 TCP extensions. These extensions collide with
     PPP/SLIP-VJC, so they should only be enabled for bus interfaces,
     not for point-to-point interfaces.

net.inet.tcp.rfc1644
     Enables T/TCP.

net.inet.tcp.mssdflt
     Sets the default maximum segment size for TCP. Usually this number
     should not be changed. It is normally not used, because Miami uses
     Path MTU Discovery to determine optimum MSS values.

net.inet.tcp.rttdflt
     This option controls TCP's retransmission timing and should not be
     changed.

net.inet.tcp.keepidle/keepintvl
     These options control TCP's keep-alive timer and should not be
     changed.

net.inet.tcp.sendspace/recvspace
     These options define the default TCP send/recv window size, and
     should usually not be changed.

net.inet.tcp.bulkftp
     Reserved for future use. Currently non-functional. Do not touch.

net.inet.tcp.initwin
     Define the number of packets in the initial TCP window for new
     connections. The default is 1, but recently research has shown
     that in some circumstances it can be beneficial to set this value
     to 2 or 3 for better performance.

net.inet.tcp.fastlocal
     Enables a new optimization that significantly speeds up conection
     to localhost.

net.inet.udp.checksum
     Enables UDP checksums for all outbound packets. This option should
     always be enabled.

net.inet.udp.maxdgram/recvspace
     These options control UDP packet thresholds and should not be
     changed.
```

```
dns.cache.size
     Controls the size of Miami's built-in DNS cache.

dns.cache.flush
     Settings this option to 1 flushes Miami's built-in DNS cache.

inetd.retrytime/toomany/cntintvl/maxbuiltin
     These options control InetD's built-in SYN flood protection. If
     you are running a very busy web server you might want to adjust
     these parameters (in particular "toomany") so clients do not get
     spurious error messages if your machine is overloaded.

dns.cache.enabled
     Enables or disables Miami's built-in DNS cache. The default value
     is 2, i.e. all host entries are cached. If this variable is set to
     1 then only host entries with a single IP address are cached, so
     interferences with round-robin IP address shuffling are avoided.
     If this variable is set to 0 then Miami's DNS cache is completely
     disabled. You should only disable the cache if you have a very fast
     connection to a local DNS server.

dns.cache.split
     This variable is usually 0, indicating that Miami uses a unified
     DNS cache for forward and reverse lookups. If you set this variable
     to 1 then Miami uses separate forward and reverse DNS caches. This
     makes diagnostic output (e.g. from MiamiNetStat) slower, but
     guarantees `correct' reverse resolution of all IP addresses (using
     PTR lookups).

inetd.retrytime
     Defines the delay after which InetD retries to bind to a socket if
     it failed to do so the first time.

inetd.toomany
     Defines the maximum number of connections InetD will accept within
     a given time interval.

inetd.cntintvl
     Defines the time interval corresponding to inetd.toomany.

inetd.maxbuiltin
     Defines the maximum number of built-in servers spawned by InetD.

inetd.processpri
     Defines the process priority for servers launched by InetD.  The
     default is -5. You should raise this value if you are running any
     CPU-intensive background processes (e.g. the RC5 challenge
     client). Otherwise your servers will never get any CPU time.

inetd.diagbufsize
     Defines the size of socket buffers for diagnostic InetD services
     (chargen, echo etc.). By default this value is 4096, i.e. smaller
     than typical UDP/TCP socket buffers, to reduce the impact of
     denial-of-service attacks.

inetd.diagtimeout
     Timeout in seconds after which connections to diagnostic InetD
     services are terminated.

socket.maxqlen
     This option defines the length of the socket connection queue for
     a listen()-parameter of 5. The default is 7, but if you are
     connected to a very fast network and have sufficient memory you
     might want to increase this value to reduce the effects of SYN
     flood attacks.
```

1.139 Miami.guide/NODE_UTILITY_TCPDUMP

```
MiamiTCPDump
============

   MiamiTCPDump allows you to dump traffic on a network after filtering
it.

   Usage: MiamiTCPDump [-adflnNOqStvx] [-c count] [-F file] [-i
interface] [-r file] [-s snaplen] [-T type] [-w file] [expression]

   Options:

-A
     Used in combination with '-x': prints packets in ASCII in addition
     to a hex dump.

-a
     Attempt to convert network and broadcast addresses to names.

-c count
     Exit after receiving <count> packets.

-d
     Dump the compiled packet-matching code in a human-readable form to
     standard output and stop.

-dd
     Dump the compiled packet-matching code as a program fragment.

-ddd
     Dump the compiled packet-matching code as decimal numbers
     (preceded with a count).

-f
     Print "foreign" internet addresses numerically rather than
     symbolically.

-F file
     Use <file> as input for the filter expression. An additional
     expression given on the command line is ignored.

-i interface
     Listen on <interface> (currently "lo0" or "mi0").  If unspecified,
     MiamiTCPDump searches the system interface list for the lowest
     numbered, configured up interface (excluding loopback).  Ties are
     broken by choosing the earliest match. Currently the result of
     this search is always "mi0".

-l
     Make stdout line buffered.  Useful if you want to see the data
     while capturing it.

-n
     Don’t convert addresses (i.e., host addresses, port numbers, etc.)
     to names.

-N
     Don’t print domain name qualification of host names.  E.g., if you
     give this flag then MiamiTCPDump will print "nic" instead of
     "nic.ddn.mil".

-O
     Do not run the packet-matching code optimizer.  This is useful only
     if you suspect a bug in the optimizer.

-p
     Do not use promiscuous mode. If an MNI driver is used then
     MiamiTCPDump by default sets the interface into promiscuous mode
     for as long as MiamiTCPDump is running. Using this option disables
     that feature, i.e.  it leaves the interface in its normal mode.

-q
     Quick (quiet?) output.  Print less protocol information so output
     lines are shorter.

-s snaplen
     Snarf <snaplen> bytes of data from each packet rather than the
     default of 68. 68 bytes is adequate for IP, ICMP, TCP and UDP but
     may truncate protocol information from name server and NFS packets
     (see below).  Packets truncated because of a limited snapshot are
     indicated in the output with "[proto]", where <proto> is the name
     of the protocol level at which the truncation has occurred.  Note
     that taking larger snapshots both increases the amount of time it
     takes to process packets and, effectively, decreases the amount of
     packet buffering.  This may cause packets to be lost.  You should
     limit <snaplen> to the smallest number that will capture the
     protocol information you’re interested in.

-S
     Print absolute, rather than relative, TCP sequence numbers.

-T type
     Force packets selected by <expression> to be interpreted the
     specified <type>. Currently known types are
        * rpc (Remote Procedure Call)

        * rtp (Real-Time Applications protocol)

        * rtcp (Real-Time Applications control protocol),

        * vat (Visual Audio Tool),
        * wb (distributed White Board).

-t
     Don't print a timestamp on each dump line.

-tt
     Print an unformatted timestamp on each dump line.

-v
     (Slightly more) verbose output.  For example, the time to live and
     type of service information in an IP packet is printed.

-vv
     Even more verbose output.  For example, additional fields are
     printed from NFS reply packets.

-w file
     Write the raw packets to <file> rather than parsing and printing
     them out.  They can later be printed with the "-r" option.
     Standard output is used if <file> is "-".

-x
     Print each packet (minus its link level header) in hex.  The
     smaller of the entire packet or <snaplen> bytes will be printed.

   <expression> selects which packets will be dumped.  If no
<expression> is given, all packets on the net will be dumped.
Otherwise, only packets for which <expression> is 'true' will be dumped.

   The syntax for <expression> is extremely comprehensive and beyond
the scope of this documenation. For a complete description of the
syntax and of the details of the output format please have a look at
the documentation for the freely distributable BSD version of
"tcpdump". Here are some examples for valid expressions:

"host sundown"
     To print all packets arriving at or departing from "sundown".

"host helios and ( hot or ace )"
     To print traffic between "helios" and either "hot" or "ace".

"ip host ace and not helios"
     To print all IP packets between "ace" and any host except "helios".

"tcp[13] & 3 != 0"
     To print the start and end packets (SYN and FIN) of each TCP
     conversation.

"icmp[0] != 8 and icmp[0]!= 0"
     To print all ICMP packets that are not echo requests/replies
     (i.e., not ping packets).
```

1.140 Miami.guide/NODE_UTILITY_TRACEROUTE

```
MiamiTraceRoute
===============

   Print the route packets take to a network host.

   Usage: MiamiTraceRoute [-m max_ttl] [-n] [-p port] [-q nqueries]
[-r] [-s src_addr] [-t tos] [-v] [-w waittime] host [packetsize]

   Options:

-m max_ttl
     Set the max time-to-live (max number of hops) used in outgoing
     probe packets.  The default is 30 hops.

-n
     Print hop addresses numerically rather than symbolically and
     numerically (saves a nameserver address-to-name lookup for each
     gateway found on the path).

-p port
     Set the base UDP port number used in probes (default is 33434).
     MiamiTraceRoute hopes that nothing is listening on UDP ports base
     <port>+nhops-1 at the destination host (so an ICMP PORT_UNREACHABLE
     message will be returned to terminate the route tracing).  If
     something is listening on a port in the default range, this option
     can be used to pick an unused port range.

-q nqueries
     Set the number of probes per "ttl" to <nqueries> (default is three
     probes).

-r
     Bypass the normal routing tables and send directly to a host on an
     attached network. If the host is not on a directly-attached
     network, an error is returned.

-s src_addr
     Use the following IP address (which must be given as an IP number,
     not a hostname) as the source address in outgoing probe packets.
     On hosts with more than one IP address, this option can be used to
     force the source address to be something other than the IP address
     of the interface the probe packet is sent on.  If the IP address
     is not one of this machine’s interface addresses, an error is
     returned and nothing is sent.

-t tos
     Set the type-of-service in probe packets to the following value
     (default zero). The value must be a decimal integer in the range 0
     to 255.  This option can be used to see if different
     types-of-service result in different paths.

-v
     Verbose output.  Received ICMP packets other than TIME_EXCEEDED
     and UNREACHABLE are listed.
```

```
-w
     Set the time (in seconds) to wait for a response to a probe
     (default 3 sec.).
```

1.141 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 three solutions to this:

* Upgrade to Miami Deluxe. It contains "MiamiTelnet", a new full-featured telnet client for AmigaOS.

* Use a different version of telnet, e.g. "AmTelnet", a nice MUI-based graphical telnet client available from www.vapor.com, the telnet version available from Aminet in comm/tcp, a terminal program together with telser.device, or "napsaterm" in telnet-mode.

* 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.

1.142 Miami.guide/NODE_RESTRICTIONS

Restrictions
****

The demo version has the following limitations:

* After 60 minutes the modem hangs up the line. SANA-II connections are interrupted after 30 minutes.

* It is not possible to keep TCP connections alive when the modem hangs up.

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

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

* Phone logging is disabled.

* The Window customization options are disabled.

* Multicasting and T/TCP are not functional.

* The IP filter is not available.

* Ping flood protection is not available.

* The sorting, merging and Clipboard import/export functions on the Database are not available.

* PPP Callback is not available.

* The packet monitoring callback (for external packet monitors like MiamiTCPDump) is not functional.

* System log events cannot be exported to syslog.library.

* The utility programs MiamiIfConfig, MiamiRemind, MiamiRoute, MiamiSysCtl, MiamiTCPDump, all multicasting tools, and the libraries miamibpf.library and miamipcap.library cannot be used.

* MS-CHAP support is not available.

The GTLayout GUI module has a few restrictions compared to the MUI version:

* Drag&drop sorting in Listviews is not available.

1.143 Miami.guide/NODE_HISTORY

History
****

Version 3.2b
: release version
  * The builtin serial driver did not work in the 68000/010 version.

Version 3.2a
: release version
  * Fixed a few typos in MiamiInit and MiamiRegister.

  * The ARexx command GETONLINETIME did not work in the demo version.

Version 3.2
: release version
  * Several changes. Please see the CHANGES file in the Miami main archive.

```
Version 3.0
     release version
        * Too many changes to list here. Many parts of the program have
          been rewritten from scratch. Keyfiles V3 are required now.

Version 2.1p
     intermediate release version
        * Last official release version before 3.0. Mostly bug fixes and
          minor changes since earlier versions.
```

1.144 Miami.guide/NODE_FUTURE

The future
****

My more immediate plans for the next Miami versions include

* A new API to handle automatic online/offline transitions controlled by clients.

* A completely new API ("ANDI") for much easier, protocol-independent access to TCP/IP functions from applications.

ISDN is another big issue. External ISDN terminal adapters are already completely supported. So is the ISDN-Master board in most modes, except in "synchronous PPP in HDLC mode". That mode requires new drivers and a new API, but people are working on it...

The only really major (intentional) limitation of Miami is that it is restricted to a single interface. A follow-up protocol stack "Miami Deluxe" is planned for later in 1998. Currently my plans for Miami Deluxe include support for multiple interfaces and probably many other functions useful for routers, such as Socks daemon support, IP masquerading and firewall functions.

There will be a discount for registered users of Miami towards a registration of Miami Deluxe.

1.145 Miami.guide/NODE_SUPPORT

Support
****

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

email
     kruse@nordicglobal.com

snail mail
     Nordic Global Inc.

```
     Attn: Holger Kruse
     PO Box 780248
     Orlando FL 32878-0248
     USA

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

mailing lists
     send "SUBSCRIBE miami-talk-ml" or "SUBSCRIBE miami-announce-ml" in
     the body of a mail to "Majordomo@nordicglobal.com".
```

1.146 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.

* Jim Szutowicz for his high-color versions of the "Miami" logo.

* Martin Huttenloher and Stefan Stuntz for their permission to use MagicWB images in Miami.

* Roman Patzner for new icon designs.

```
* Olaf Barthel for gtlayout.library and help in tracking down some
  problems.

* all users who decide to register Miami.
```