Fitrix Visual Menus Technical Reference Version 5.20.02

August 01, 2007

Document Id: FX-T-VM-TR

Copyright

Table of Contents

1 Introduction

Introduction

Visual Menus (**VM**) is part of the Visual Development Tools (**VDT**) Product Suite developed by Fourth Generation Software Solutions, Corp. This product offers the menu-level control necessary to create a complete, user-friendly application front-end. The menu system it creates is attractive, easy-to-create, easy-to-use, and completely portable. Menus created with Fourth Generation VM can be used to access other menus, run programs or reports on the system, or execute UNIX operating system commands, providing an alternative to the classic Fourth Generation CASE Tools character based menus used now. VM utilizes the following functionality:

- Graphical User Interface (GUI)
- Mouse Functionality
- Right-click Capability
- Drop-down Menus
- CYMK Color Configuration

VM works with all systems built on the Genero **BDS** (Four J's **B**usiness **D**evelopment **S**uite) platform. Although this product was created for use with Fourth Generation's Visual Development Tools Product Suite, it is highly adaptable for use with all software applications using Genero **BDL** (Four J's **B**usiness **D**evelopment **L**anguage).

Features

VM instructions (refer to "The VM Instruction Set") can perform any of the following specific functions:

- Call up other menus.
- Create a list of explanatory text that can be used by later instructions to show the user comments about the actions that are to be taken.
- Pause and show the user explanatory text and control the method of report output.
- Call up a report program
- Call up a file maintenance program.
- Test for the successful completion of any item instruction (particularly those that execute programs or UNIX commands).
- Change or set an environment variable.

How It Works

The VM system allows the user to invoke a set of Menu Instructions from a single menu item. Every option on a menu has a corresponding set of Menu Instructions — these instructions form a Control Language.

At the simplest level, VM allows the system administrator to quickly modify a menu. This capability goes beyond look and feel changes. For instance, a system administrator can issue permissions to a User or Group with as much access to the system as is appropriate without having to deal with the UNIX shell or the shell language.

Other menu systems may allow you to create menus that execute programs from a menu item, but VM provides you with the ability to control the execution of a "multiple-step" menu item without batch or shell programming. For instance, one menu item may control an entire series of events: running a program or series of related programs, testing for proper execution of each program, and so on. VM is more flexible, more manageable, and more portable than a UNIX shell script.

Menu Structure

This section will briefly describe the structure of a VM system.

For VM to function properly, a specific table structure is required. This structure is hierarchical and uses the ‘file folder’ metaphor familiar to Windows users. For the FG menu system the folder corresponds to a **menu** and provides a way of branching. The file corresponds to a set of instructions that are to be executed by the host, see Figure 1-01.

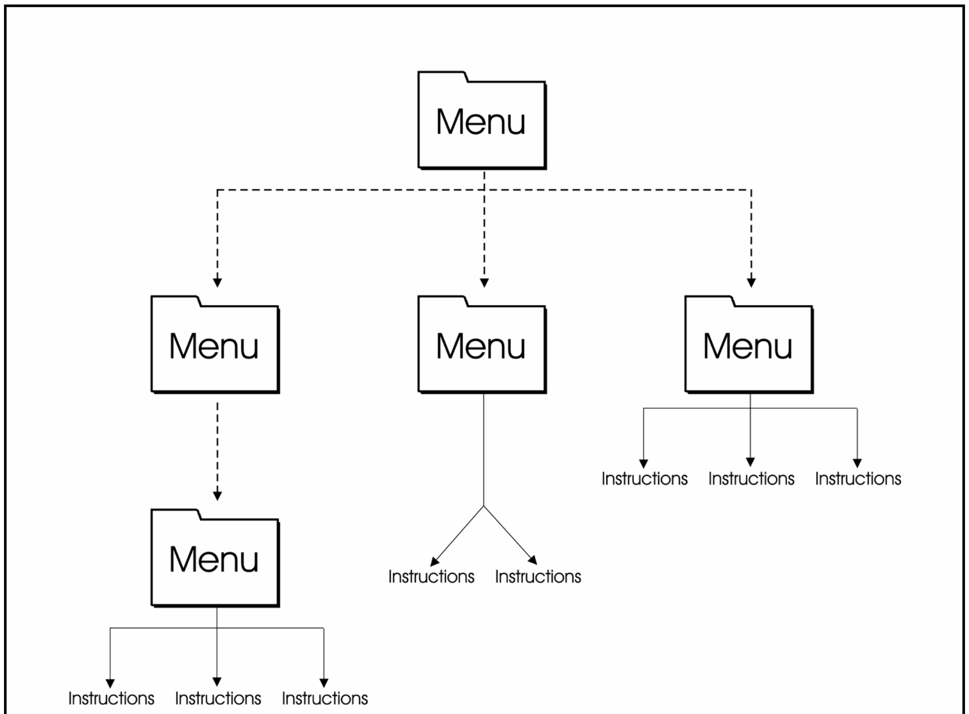


Figure 1-01

Pre-Installation Considerations

VM is included with the Visual Development Tools Product Suite. By upgrading to VDT Suite, VM will be installed automatically.

If you are a current Fourth Generation customer running Classic CASE Tools including the ‘mz’ menu product, you must run a conversion utility to read the old flat files and load them into several tables needed by VM. This is covered in detail in the Installation Section of this document.

If you are using another application system with the GENERO BDL tools and wish to enjoy the user-friendly versatility of the Menus product, you can set up a menu structure from scratch. See, Setting Up the Menu Structure, in section 3 System Administrator’s Guide, of this manual.

2 Installation

- Installation Introduction
- UNIX System Installation Environment
- PC Workstation Installation
- Installation Troubleshooting

Installation Introduction

Before beginning the installation process, make sure that you have set up Visual Development Tools correctly.

The Menus product is composed of three components:

1. A set of UNIX programs: ‘mn’, ‘mnc’, and ‘mnl’.
2. A Workstation program: ‘mntk.exe’ that is automatically executed from the UNIX side.
3. A Workstation based configuration program: ‘mnconfig.exe’.

Unix System Installation Environment

In addition to the normal Fourth Generation environment settings—the following environment variables are required:

- **APPSERVER** – IP address or name of the application server (where the menus are installed).

  `APPSERVER = Orion` (Unix server name)

  `APPSERVER = 132.147.160.101` ("Dotted" IP address)

- **FGLSERVER** - The name of IP address where the GENERO BDL display server is running.

  If a WTK workstation client running 'wtk.exe':
  ```
       $ FGLSERVER=myworkstation     ;export FGLSERVER
  ```

  If an X-Windows Unix host running 'fglX11d' deamon:
  ```
       $ FGLSERVER=theunixbox   ;export FGLSERVER
  ```

- **remoteshell** – Name of the remote shell used by the platform ("rcmd" for SCO; rsh for Linux...)

- **mn_xserver --** Set to "WTK"

- **filedir** -- Set to directory where reports are to reside.

- **lpflags** -- Switch that controls number of copies ("`-n`" for SCO, "`-#`" for Linux...)

Environment variables '$fg', '$INFORMIXDIR', and "$FGLDIR" should be set, and '/fitrix/bin', '$fg/bin', '$fgtooldir/bin', '$INFORMIXDIR/bin' and "$FGLDIR/bin" should be on '$PATH'.

Additional Unix environment variables:

IBM-Informix (IN ALPHABETICAL ORDER):

FGLDIR – GENERO BDL product install directory **(required)**
FGLGUI – (1=graphical interface on) **(required)**
FGLSERVER – IP-address or name of display server **(required)**
INFORMIXDIR – Informix product install directory **(required)**
INFORMIXSERVER – Informix DB server **(required)**

Fourth Generation (IN ALPHABETICAL ORDER):

APPSERVER – IP-address or name of applications server **(required)**
company – Database name **(required)**
fg - FGSS products root. Directory **(required)**
filedir – Directory for temporary report output **(required)**
HOME – Home directory **(required)**
ifxproject – Applications root directory (e.g. $fg/accounting) **(required)**
LPDEST – Default printer **(required)**
lpflag – Switch that controls number of copies **(required)**
printerlist – Full path to printer list **(required)**
remoteshell – Either ‘rsh’, ‘rcmd’, or ‘remsh’ **(required)**
report_pager – “fglpge.sh” for graphical pager.
SHELL – Path to current shell

Visual menu specific environment variables (IN ALPHABETICAL ORDER):

APPSERVER - IP address or name of application server
mn_analyze - Set to '1','2', '3' or '4' to turn on verbose logging. Useful for more extensive debugging.

0 (default): Minimum logging.
>0: mn,mnc,mnl-- Additional logging.
>1: mn-- Log all Menus, Security, and Messages uploaded to client.
>2: mn-- Log Menus uploaded to client.
mnc-- Log individual arguments to menu commands (:ifxreport:...).
>3: mn-- Log all traffic uploaded to client.

mn_applications – System administrator imposed limit as to number of applications that a user can launch (default is 4).
mn_buttonbar – Set to ‘0’ if button bar should not appear, set to ‘1’ if should appear, set to ‘2’ if additional shell button appears (default is ‘1’).
mn_company - Where 'menucvt.sh' is to load the menus from (default '$company')
mn_company_list - Colon separated list of valid companies where menus may exist (default is $company).
mn_editor - Local (PC workstation) editor (defaults are: $FGLEDITOR then 'wordpad.exe')
mn_execorder – Colon separated list of extensions controlling search order of program to execute (default: 42r:4gi:4ge).
mn_fglpge_132 - Does the screen pager use 132 columns (default=1).
mn_hide_deny_options - Set to '1' to hide rather than to gray out options denied to a user-- (default: '0'),
mn_installdir – Visual Menu installation (if other that the <WTK/GDC-install> directory).
mn_keepconnected - Keep connected to the database throughout the VM session (default=0).
mn_language - Language (Default="ENG").
mn_logintitle - Title of Login Terminal Window (used when shelling out to control the terminal window.
mn_maintitle – Title in Visual Menu title bar.
mn_menubar – Set to ‘0’ if menu bar should not appear, set to ‘1’ if should appear, set to ‘2’ if only File-Exit button appears (default is ‘1’).
mn_menu_company - Database to load menus from (default=$company).
mn_mnconfig – Set to ‘1’ if button bar should show ‘Launch Visual Menus ConfigurationTool’ button (default: ‘0’).
mn_modify – Global to [dis]allow modification of reports; Set to ‘e’ to disallow modification, set to ‘E’ (or unset) to allow modification. NOTE: use of ‘e’ or ‘E’ with the pause command on a menu selection overrides this setting
mn_modline - When editing reports the string to add to the top of a edited report (default is unset which uses the text “<This report has been modified.>”).
"" (empty string): Nothing is printed.
"*Custom text*"
mn_noclient - Set to '1' to not bring up the workstation part automatically (mntk.exe by **$remoteshell**).
mn_nolocalprinter – Set to anything if local printer suppressed (default is unset)—see switch -l.
mn_nomail - Set to anything if mail suppressed (default is unset) —see switch -m.
mn_pipedir – Where temporary pipe files reside (default “/tmp”).
mn_noreprint – Set to ‘1’ to disable reprinting (close printer dialog after initial print selection)-- (default “0”) .

mn_reportprinter - Deprecated (same as mn_noreprint).
mn_reportsave - Global-- save all reports.
mn_sessions - Number of open VM sessions (default=1).
mn_socket_retry - Maximum attempts for host to connect to client (default=10)
mn_socket_timeout - Seconds to wait for each socket attempt (default=8).
mn_xserver - Type of client -- WTK (MS Windows) **(required)**
mz_nofax - Set to anything if fax suppressed (default is unset) —see switch -f.

• PC Workstation Installation

- ✓ Visual Menus will not automatically begin the installation process upon inserting the CD into the CD ROM Drive. Follow these steps:

  - Insert the CD into the drive.
  - For a Windows PC, click Start, then Run. The Run dialog box displays and prompts you for a program file.
  - Browse the CD for the Workstation folder.
  - Open the folder and browse for the install.exe program.
  - Select vminstall.exe, then click OK on the Run dialog box. The installation procedure will begin.

- ✓ During the installation procedure, you will be prompted to specify a location. You must install Menus into the same directory that the FourJs Client product was installed in.

- ✓ During installation, you may be asked whether to overwrite existing files. Answer 'Yes to all'.

Installation Troubleshooting

✓ ***What files are on the Installation Media?***

UNIX Host

The file "/mnt/cdrom/server/media/<platform>/mn.tgz" (extracted during installatio
contains the files under $fg.

```
To $fg/bin--
        mn, mnc, mnl, mncv, mndiag, mntk
        mn_rcp_from_pc.sh, mn_rcp_to_pc.sh
        menucvt.sh, mn_schema_chg.sh
        fglicen, fglilook
        fglpge.42f, fglpge.42m, fglpge.42r, fglrun_pge, fglrun_pge_914, fglpge.sh
        fg_licen
        mz_mn
To: $fg/fgss_bin
        brown.res, default.res, fg.res, green.res, plum.res, sepia.res, user.res
To: lib/workstation/bmp/base
        fglprofile_pge
And others...
```

Workstation Host

- **The following files are copied during installation on the Workstation host:**

```
To:  <WTK_install_dir>/fgss_bin--
        mntk.exe mnconfig.exe
        tclsend.dll winmgr.dll winmon.dll wndexec.dll tcl83.dll
        brown.res default.res fg.res mn.res green.res plum.res sepia.res
And others...
```

```
To:  <WTK_install_dir>/bmp--
        fgsplash.bmp logo.bmp tl_copy.bmp tl_cut.bmp tl_paste.bmp tl_bof.bmp tl_eof.bmp
        eyedrop.cur
        mntklogo.gif
And others...
```

✓ **How to run a GENERO BDL program from within another GENERO BDL program:**

Let us say that you effectively want to run the following commands:

```
# fglrun o_termls.4gr -c sample filter 'strtermr.due_days=45'
# fglrun o_termls.4gr -c sample filter "strtermr.due_days=45"
```

Example 1:

The menu command would be set up as follows:

```
:ifxreport:ar:o_termls:$chg_filter:default:::
```

Code would be:

```
LET cCmd="(chg_filter=\"filter 'strtermr.due_days=45'\";
export chg_filter;mz -i e armenu customer)"

LET cCmd="(chg_filter='filter\"strtermr.due_days=45\
"'; export chg_filter;mz -i e armenu customer)"
```

Example 2:

The menu command would be set up on one line as follows:

```
:ifxreport:ar:o_termls:filter $chg_filter:default:::
```

Code without be:

```
LET cCmd="(chg_filter='strtermr.due_days=45'; export
chg_filter;mz -i e armenu customer)"

LET cCmd="(chg_filter=\"strtermr.due_days=45'\"; export
chg_filter;mz -i e armenu customer)"
```

✓ ***Log files?***

Logs are written to aid in diagnosing problems. On the Unix host side they are: $HOME/mn_p_nnnn.log (or .bak) and $HOME/mn_c_nnnn_nnn.log (or .bak), on the workstation side it <WTK_install_directory>/fgss_bin/mntk.log

> There is always a possibility that a particular login will be used simultaneously more than once. To accommodate for this, a four digit numeric is used in the log file name. When the log is opened, an available name is selected: mn_p_nnnn.log or mn_c_nnnn_nnn.log. When the user's session is normally exited, it is renamed mn_p_nnnn.bak or mn_c_nnnn_nnn.bak so that it is free to be reused. Therefore, if a file exists in the user's home directory named mn_p_nnnn.log or mn_c_nnnn_nnn.log , it either means that a session is currently in use or that a session terminated abnormally. If the latter, the log file can be analyzed to see what has happened.

✓ ***Where does the Visual Menus get the runner used with the pager?***

The runner used by the menuing system is a link in: `'$fgtooldir/bin'` pointing to `'$FGLDIR/bin/fglrun'`. The module `'fglpge.so'` is dynamically linked according to the value of `'$FGLLDPATH'` which is set to: `'$fglibdir/lib'.`

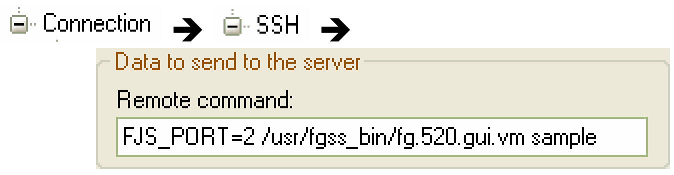
✓ ***I already have used the earlier 'mz' product, can I still use it for 'dumb' terminals?***

No

What is happening behind the scenes between the Visual Menus and GENERO BDL?

The user brings up Visual Menus using the 'PuTTY' client. The critical PuTTY settings are:

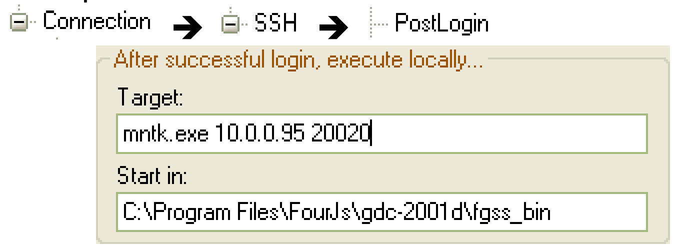
- Host piece:



Remote command: FJS_PORT=2 /usr/fgss_bin/fg.520.gui.vm sample

The script here (whatever it's name) brings up the host piece (mn).

- Workstation piece:



Target: mntk.exe 10.0.0.95 20020

Start in: C:\Program Files\FourJs\gdc-2001d\fgss_bin

This launches the workstation piece (mntk.exe).

Program '**mntk.exe**' on the workstation host now makes a socket connection to '**mn**' on the Unix side and they begin transfering information.

First the menu structure is uploaded to the workstation and the menu's are painted. Each time the user 'clicks' a GENERO BDL program execution option, a message is sent back to the Unix host which then executes the GENERO BDL program.

- New network service

Modify the Unix networking file '/etc/services' and provide a port for the system. At a Unix prompt, use 'vi' to add an entry.

```
$ vi /etc/services
```

```
fgmn    20XX0/tcp    # FG Menu
```

Customer Support

For assistance, contact Customer Service at:

Fourth Generation Software Solutions
Voice: (800) 374-6157
FAX: (770) 432–3448
support@fourthgeneration.com

3 System Administrator's Guide

- Setting Up the Menu Structure
- Creating Users and Groups
- Assigning Permissions
- Menu Control Language

Setting Up the Menu Structure

Setting up a menu structure begins by initializing the appropriate tables for your company. You can manually enter data into the tables through the Menu system, or load them from a previous version of the 'mz' system.

Manual Data Entry

1) Initialize the tables:

```
$ company=standard ;export company
$ menucvt.sh -I mainmenu
```

The menu name must use lowercase alphanumeric characters plus '_' and '.' (dot). The tables 'cgsmnitm', 'cgsmncmd', and 'cgsmnsec' are created.

2) Start Fourth Generation Menus. The following menu displays:

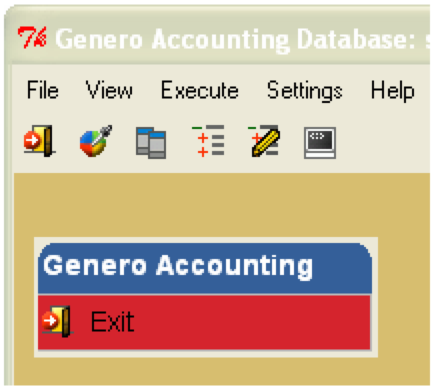
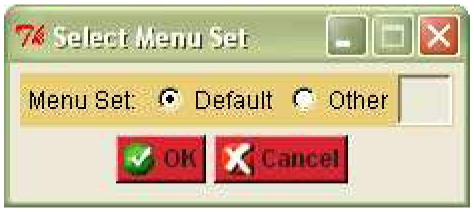


Figure 3 - 01

The menu structure is currently empty. In order to populate the menu structure you must switch to 'Explorer View'. This view allows you to view folders as you create them.

3) Click the ***Edit View*** icon:



4) Click:

5) The following appears:

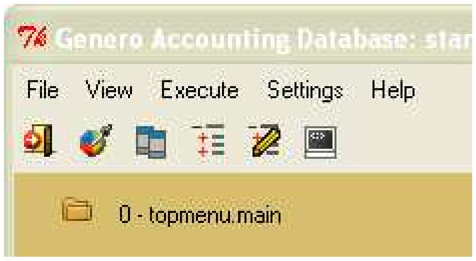

Genero Accounting Database: star
File View Execute Settings Help
0 - topmenu.main

Figure 3 - 02

The folder labeled "topmenu.main" is a placeholder for the top or root of your menu system.

6) Right-click on 'topmenu.main' title:

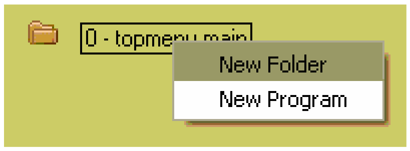
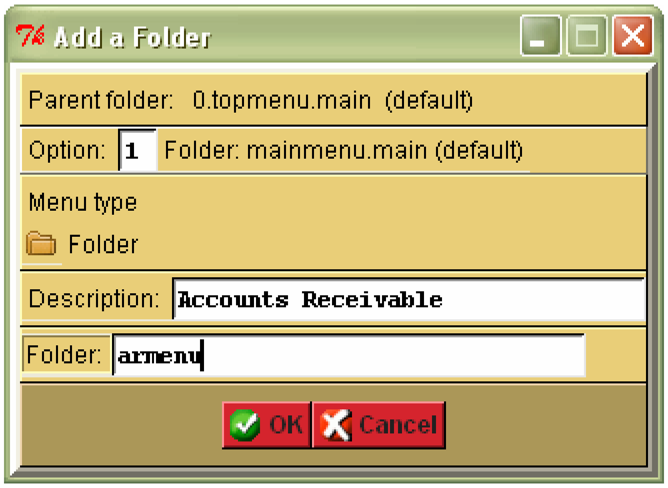


Figure 3 - 03

Click:
.

The 'Add a Folder" dialog box displays:



Parent folder: 0.topmenu.main (default)

Option: 1 Folder: mainmenu.main (default)

Menu type

Folder

Description: Accounts Receivable

Folder: armenu

Figure 3 - 04

Figure 3 - 05 below shows how the entries in the Edit a Folder dialog box are used in the menu system. This example shows a menu system categorized using numerals, however you can use letters. You should have a plan for the structure of your menu system before continuing.

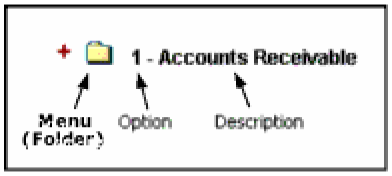


Figure 3 - 05

The following fields require information to create/edit a folder.

- Option – This is an alphanumeric field and allows you to label levels in the file system
- Menu type (controls icon) – FL = Menu folder
  SC = Screen
  RP = Report
  OT = Other
  PR = Process
  HD = Hidden

- Description – Title that will be displayed in the Folder View
- Folder - Name of the folder

When complete, click OK on the dialog box. The screen refreshes and the new folder has been added under the 'topmenu.main' folder.

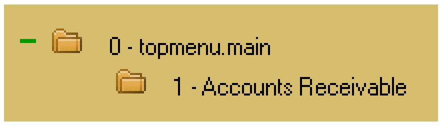
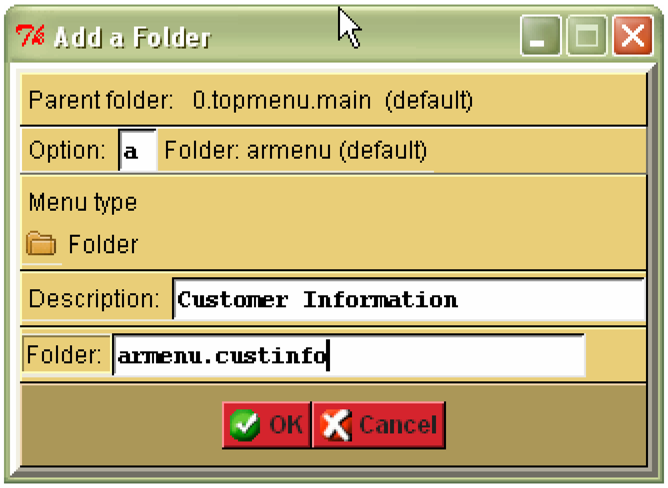


Figure 3 – 06

Repeating the process by right-clicking:

and then:
.



Parent folder: 0.topmenu.main (default)

Option: a Folder: armenu (default)

Menu type

Folder

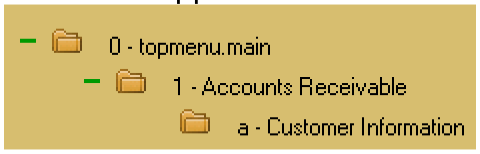
Description: Customer Information

Folder: armenu.custinfo

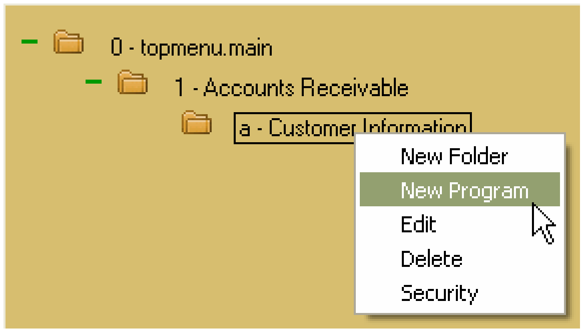
Click

on the dialog box.

A new sub-folder appears:

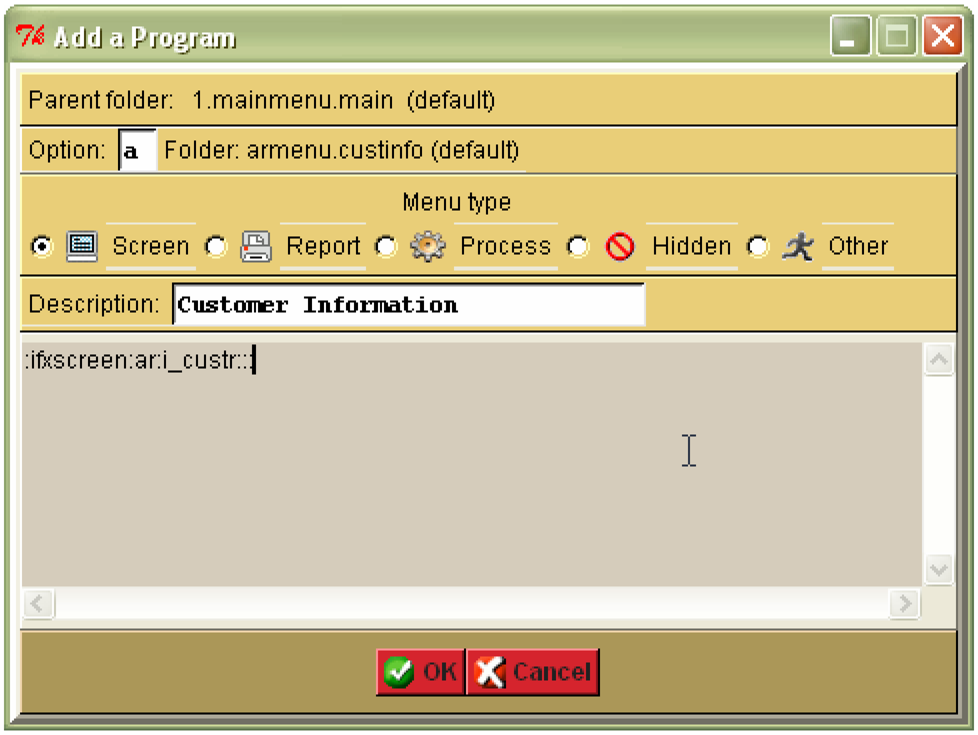


To create a program, right click on:



and click:

The ‘Add a Program’ dialog box appears:



Parent folder: 1.mainmenu.main (default)

Option: a Folder: armenu.custinfo (default)

Menu type

[✓ Screen] Report Process Hidden Other

Description: Customer Information

```
:ifxscreen:ar:i_custr::
```

Figure 3 - 07

Figure 3 - 08 below shows how the entries in the Add a Program dialog box are used in the menu system. This example shows a menu system categorized using letters, however you can use numerals. You should have a plan for the structure of your menu system before continuing.

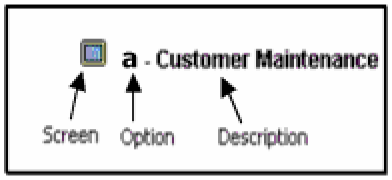


Figure 3 - 08

The following fields require information to create/edit a folder.

- Option – This is an alphanumeric field and allows you to label levels in the file system
- Menu type – SC (creates a screen)
  {Other options are: FL = Folder; RP = Report; OT = Other}
- Description – Title that will be displayed in the Folder View
- Instruction area – Instructions to be carried out, see Menu Control Language on page 11.

Click: OK

A new program appears:

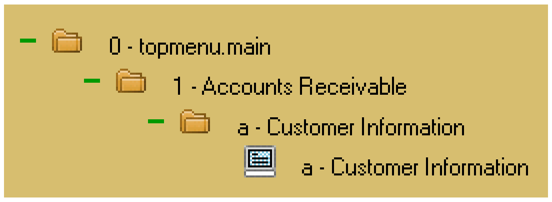


Figure 3 - 09

Clicking the ‘+’ sign in front of a folder icon expands the folder, displaying the list of the files within the folder. The ‘-‘ sign collapses the list.

The text scrollable area in the ‘Add a Program’ dialog box that contains the text “**ifxscreen:ar:i_custr:::**” is the “Control Language” that determines what happens when a user clicks an item. It is critical that this code is accurate for VM to operate correctly. See the Menu Control Language section on page 11 for more information.

Creating Users and Groups

Security for the menu system is based upon two tables, one that contains users (stxsecur) and one that contains groups (stxgropr) There is an additional table (cgsmnsec) that controls which user/group is allowed or disallowed to execute a menu item. Maintenance of users and groups is accomplished through GENERO BDL programs launched from the Menu Bar while maintenance of security for a particular menu item is controlled by right clicking on the item from the Explorer View.

The database administrator must first assign permissions to users who need full access to the Menus system. After that, all of the Groups and Users who will use the Menus system need to be defined.

To manage users, select the Execute dropdown menu, select Security, then click: User and Group Permissions.

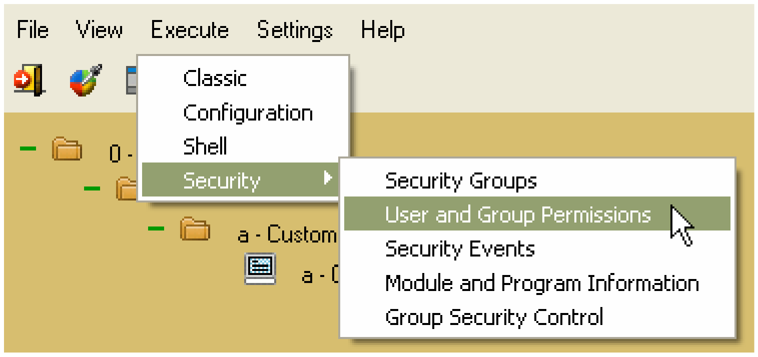


Figure 3 - 11

The screen to Manage User and Group Permissions displays.

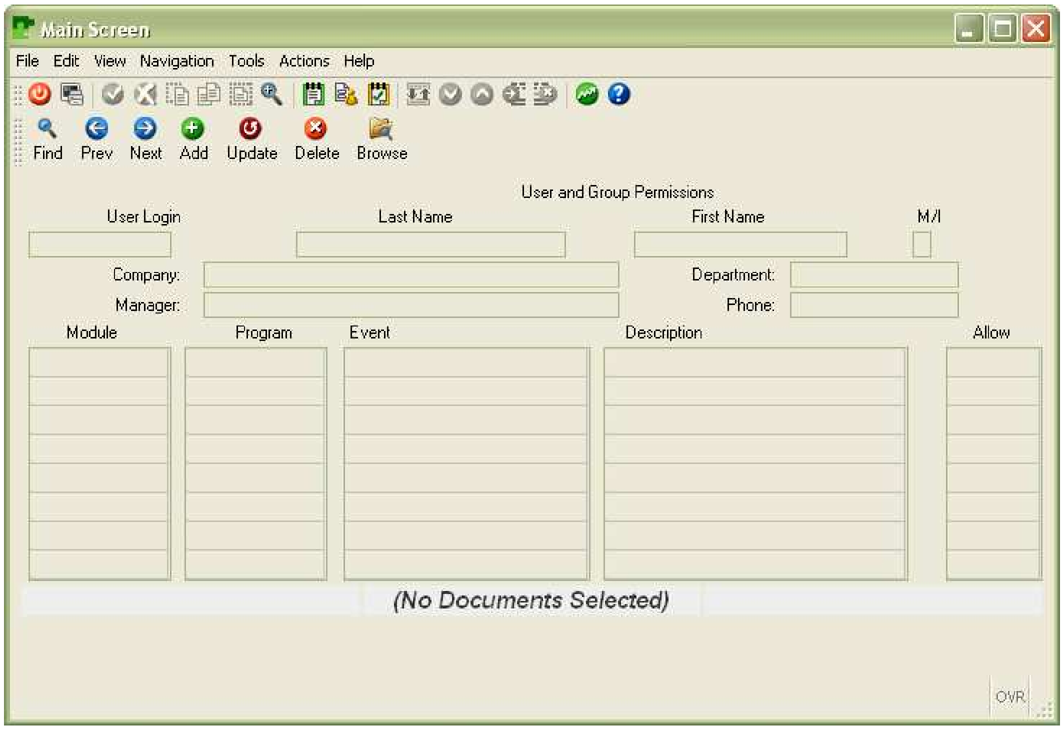


Figure 3 - 12

Establish a new User by entering a User Login, and then fill in optional fields of name, company, manager, department, and telephone fields. When complete, click the Accept button on the right-hand side of the screen. The information is submitted to the database, and the User is now allowed to use the Menus system.

You will also want to establish Groups of Users who have the same permissions to access Menu items. From the Execute dropdown menu select Security, then click: Security Groups. (See Figure 3 – 13).

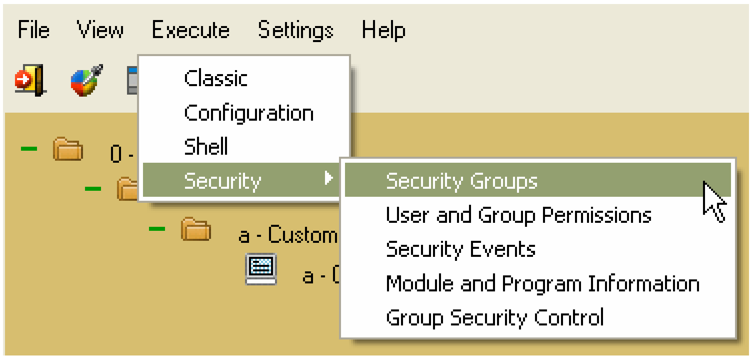


Figure 3 - 13

The Manage Security Groups screen displays. Fill in the group code, description and user logins.

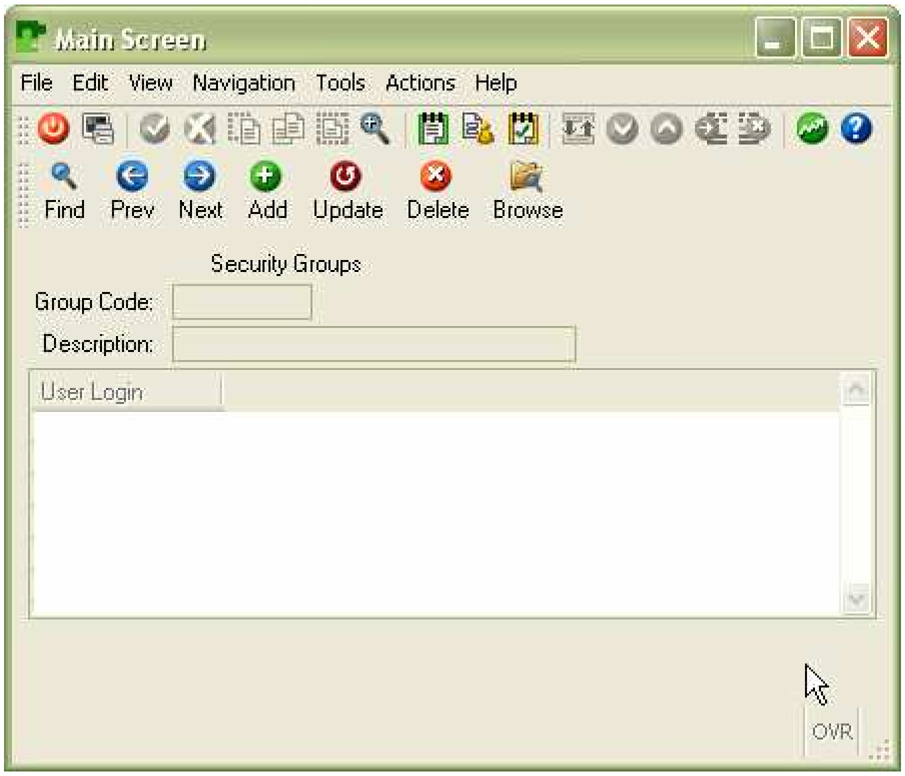


Figure 3 - 14

From the Execute dropdown menu select Security, then click: Security Groups. (See Figure 3 – 15).

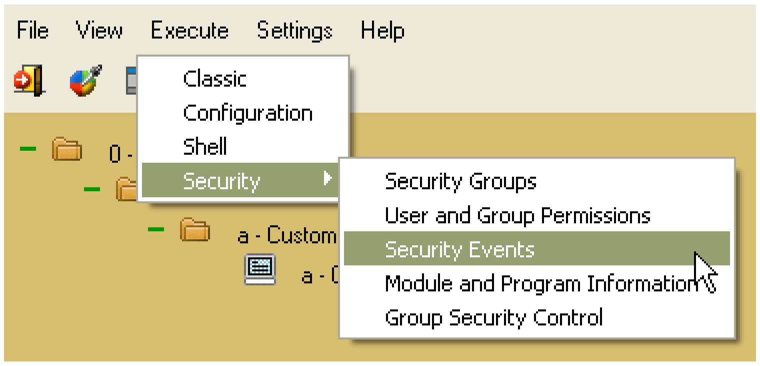


Figure 3 - 15

The Manage Security Events screen displays.

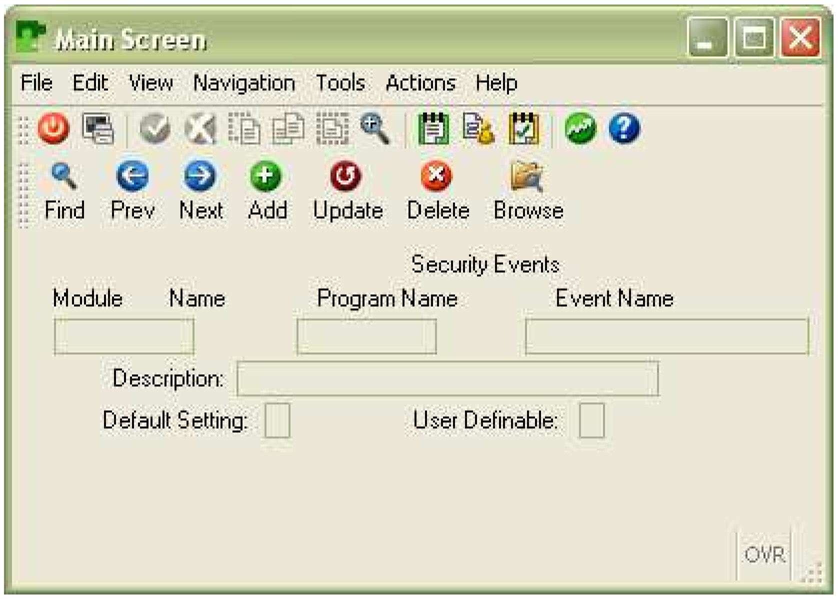


Figure 3 - 16

From the Execute dropdown menu select Security, then click: Module and Program Information. (See Figure 3 – 17).

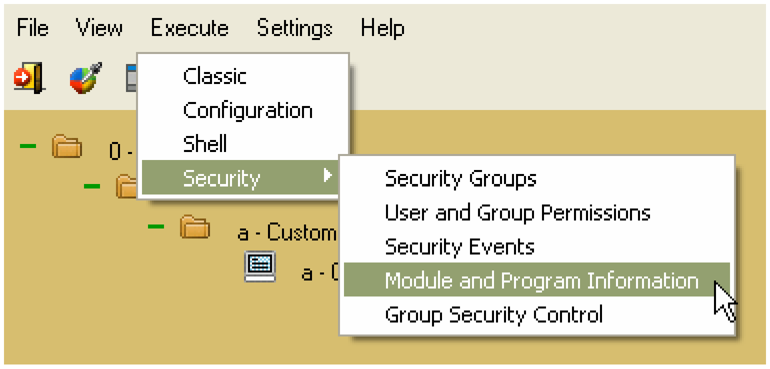
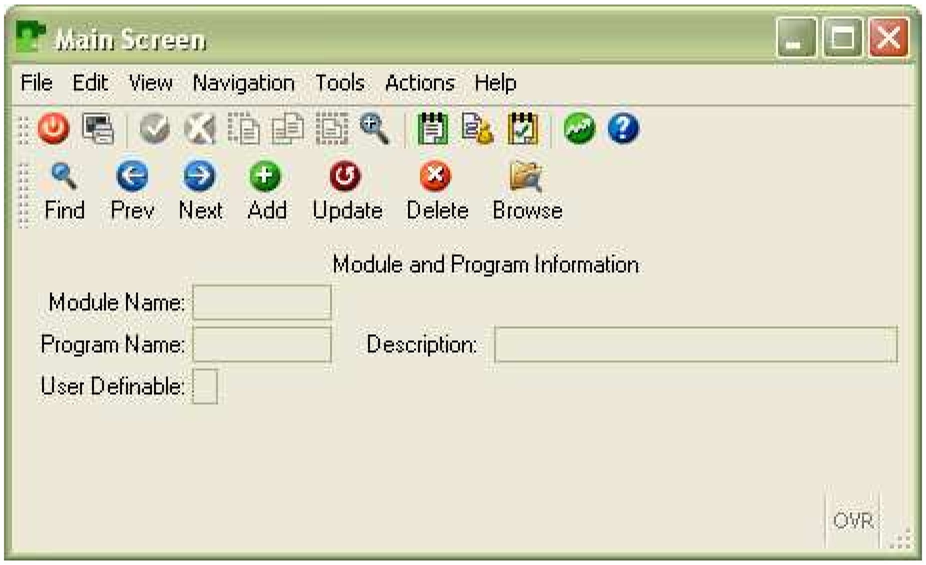


Figure 3 - 17

The Manage Module and Program Information screen displays.



Module and Program Information

Module Name:

Program Name:

Description:

User Definable:

Figure 3 - 18

From the Execute dropdown menu select Security, then click: Group Security Control. (See Figure 3 – 19).

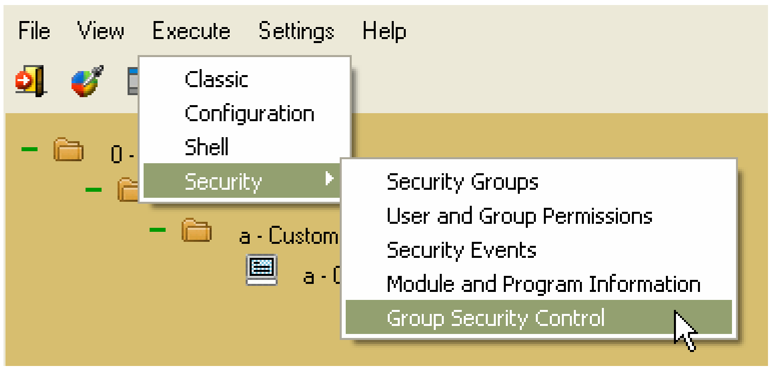
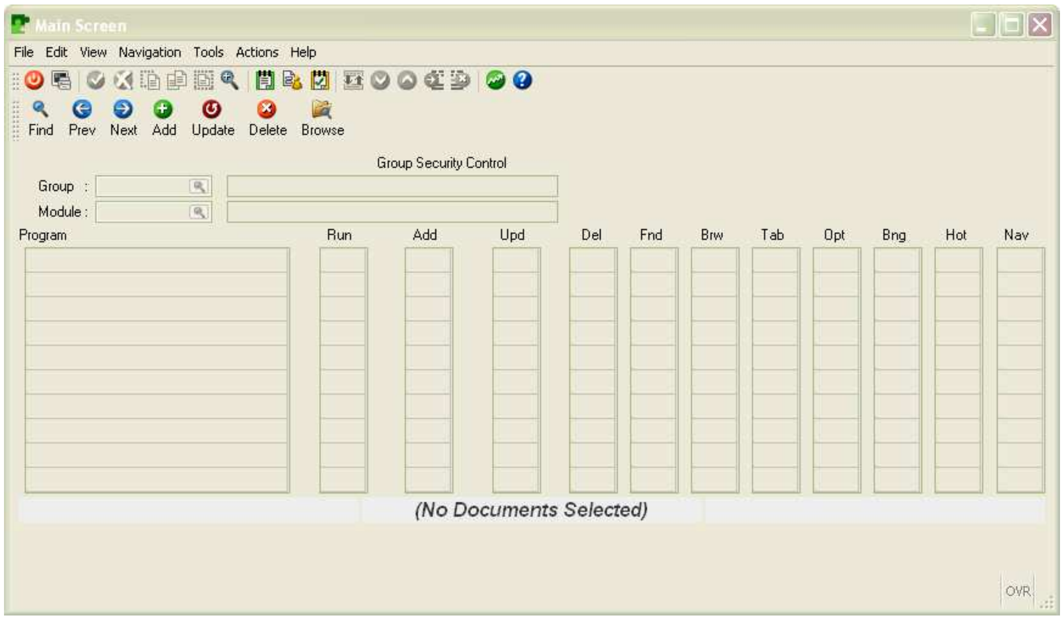


Figure 3 - 19

The Group Security Control Information screen displays.



Group Security Control

Group :

Module :

Program	Run	Add	Upd	Del	Fnd	Brw	Tab	Opt	Bng	Hot	Nav

(No Documents Selected)

Figure 3 - 20

Assigning Permissions

Security for the menu system is based upon two tables, one that contain users (stxsecur) and one that contains groups (stxgropr) There is an additional table (cgsmnsec) that controls which user/group is allowed or disallowed to execute a menu item. Maintenance of users and groups is accomplished through GENERO BDL programs launched from the Menu Bar while maintenance of security for a particular menu item is controlled by right clicking on the item from the Edit View.

To assign security permissions, switch to 'Edit View' and right click on any item (folder or otherwise). Select 'Security' from the pop-up menu.

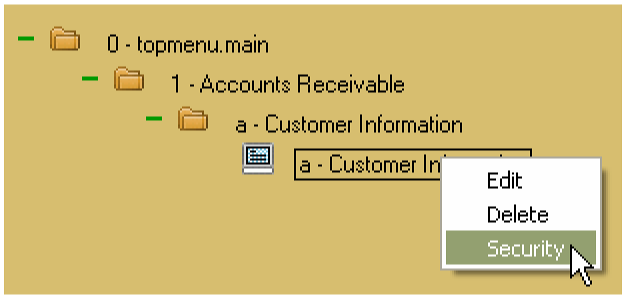


Figure 3 - 21

The ‘Maintain Security’ Dialog Box will appear. The user has clicked on the group ‘Payroll’ in preparation for assigning it.

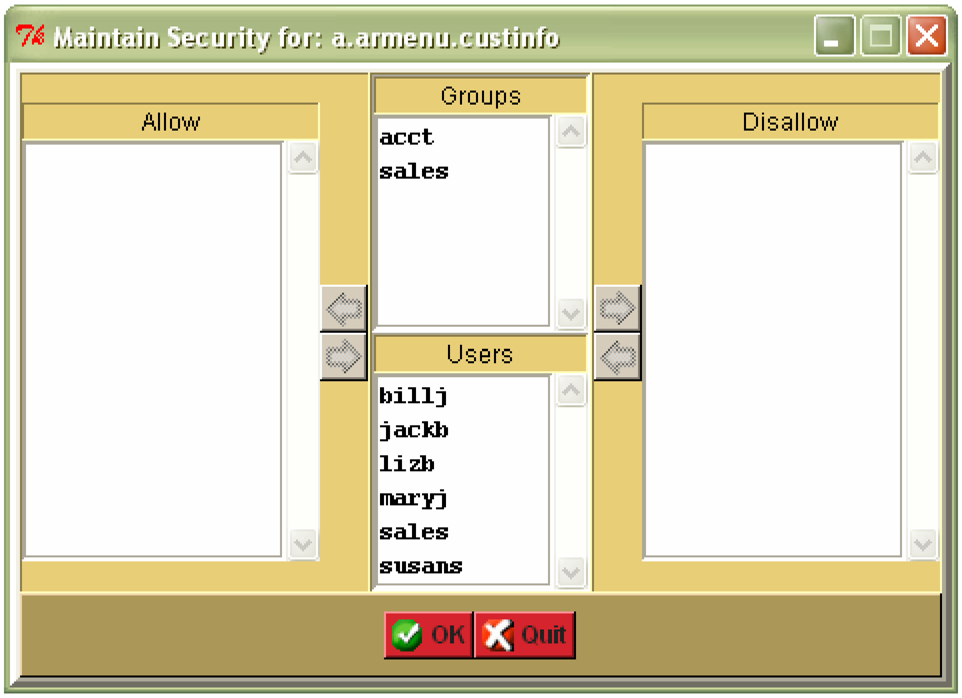


Figure 3 - 16

In the center you will see a scrollable list of groups (top) and a scrollable list of users (bottom). The group ‘payroll’ is highlighted after being clicked and the top arrows have been highlighted and the lower arrows have been dimmed since the only possibilities are to move the group either to the ‘Allow’ (left) or ‘Disallow’ (right) column.

Assume the user clicks on the upper left arrow (move to ‘Allow’). This will move the group to the ‘Allow’ list. Now assume that the user has clicked on ‘payroll *’ again (the asterisk is an indicator that the name was a group).

The user may now return the group to its original position (out of the ‘Allow’ list and back to the ‘Groups’ list).

Menu Control Language

Each line of Control Language contains an instruction which always begins and ends with a colon (`:`). Following this initial colon is the instruction name, followed by a colon, followed by the instruction parameters, which are separated from each other with colons. For example:

```
:ifxscreen:ar:i_custr:::
```

A VM set of item instructions may consist of any number of instructions and comments. Both the instruction and its parameters **must** be written in lowercase (except for parameters consisting of text that is to be displayed, which may appear in upper or lowercase). Some of these instruction parameters are required, whereas others are optional. After selecting a menu item, processing begins. If an instruction is missing a required parameter, the system reports a format error.

Menu Control Language Instruction Set

VM has two special instructions designed to assist developers working in an GENERO BDL environment. These instructions are the root of most sets of item instruction. They allow you to run a specific GENERO BDL report or a program.

- The **`:ifxscreen:`** instruction searches a directory structure for a specified program, passes arguments to that program, and runs the program.
- The **`:ifxreport:`** instruction simplifies the entire process of selecting reports to be run, passing arguments to those reports, and routing the output to a specific destination.

The `:ifxscreen:` Instruction

The `:ifxscreen:` instruction searches a directory structure for a specified GENERO BDL program, passes arguments to that program, and then runs the program. The format:

```
:ifxscreen:module:program:[flags]:[x]:
```

An example:

```
:ifxscreen:ar:i_invce:::
```

Explanation:

module: An environment variable, $ifxproject, points to a directory in which module directories reside. The name of the module directory is the value given in the "module" field with a ".4gm" extent. In the example, "ar.4gm" is a module (directory) containing accounts receivable directories.

program: Two directories may exist below the module directory. The names of these two directories are the value given in the "program" parameter with a ".4gc" and ".4gs" extension. Both directories may exist. VM searches the .4gc directory first for the program to be run. Either the .4gc or .4gs directory can contain an executable program. The name of this program is the value given in the "program" parameter with a ".42r" extension. The program is executed with a ‘runner’ or ‘interpreter’ for GENERO BDL programs.

flags: Any "flags" set are sent to the program being called.

> **x:** When used, if an abnormal exit occurs during processing, Menus returns the user to the menu, rather than proceeding to any remaining instructions within the set of item instructions.

The `:ifxreport:` Instruction

Menu options that print reports utilize the `:ifxreport:` instruction to find and print reports. The `:ifxreport:` instruction searches a directory structure for a specified GENERO BDL program, passes arguments to that program, runs the program, and routes the output to a designated destination.

The format:

```
:ifxreport:module:program:[flags]:[destination]:[x]:
```

An example:

```
:ifxreport:ar:p_invpst:-p:default:x:
```

Explanation:

module: An environment variable, $ifxproject, points to a directory in which module directories reside. The name of the module directory is the value given in the "module" field with a ".4gm" extent. In the example, "ar.4gm" is a module (directory) containing accounts receivable directories.

program: Two directories may exist below the module directory. The names of these two directories are the value given in the "program"

parameter with a ".4gc" and ".4gs" extension. Both directories may exist. VM searches the .4gc directory first for the program to be run.
Either the .4gc or .4gs directory can contain an executable program. The name of this program is the value given in the "program" parameter with a ".42r" extension. The program is executed with a ‘runner’ or ‘interpreter’ for GENERO BDL programs.

flags: Any "flags" set are sent to the program being called.

destination: The "destination" field may be set to "screen," or "default." The "default" destination may be set with a prior `:pause:p:` instruction; otherwise, the default printer is used.

VM passes the arguments "destin /tmp/ifx*uniquenumber*" to an Informix report program. In order for you to print or redirect properly from Fourth Generation Menus, your program must parse for these command line arguments. This can easily be accomplished utilizing a call to the get_arg() function in the $fg/standard.4gs library, which is included with screen . If you do not have this library we have listed the function below for your convenience.

The following code shows an example of how you might set your program up to output its report to the filename where VM expects to find output. First, your program must read the filename from the command line, the -c $company variable, and then output to that filename.
The following function call puts the filename into a variable called destin_variable. The function get_arg() previously placed the filename in the scratch variable.

```
        if get_arg("destin") then
           let destin_variable = scratch
        end if
```

The following function outputs your report to the destin_variable:

```
##########################################################
function start_rpt()
#########################################################

start report some_report to destin_variable
   return
end function
# ct_start_rpt()
```

x: When used, if an abnormal exit occurs during processing, VM returns the user to the menu, rather than proceeding to any remaining instructions within the set of item instructions.

The `:ifxreport:` instruction prints any partial report that was generated even if it exits with a non-zero exit status.

The `:env:` Instruction

The :env: instruction is used to set variable values or to remove variables from the environment.

Format:

```
        :env:variable:[value]:
```

Examples:

```
        :env:myname:root:
        :env:pst_status:$mz_status:
        :env:pst_status::
```

In the first example, the variable myname is set to "root". Any subsequent reference to $myname uses this value. In the second example, the variable pst_status is being set to the current value of ‘mz_status’. In the third example, the variable pst_status is being removed from the :environment. A variable set with the :env: instruction is available to all subsequent programs called by Menus.

The `:exit:` Instruction

The `:exit:` instruction halts the processing of the Item Instruction Set and returns the user to the menu. It is usually used in conjunction with `:if:` statements to cut off processing if certain conditions exist.

Format:

```
        :exit:
```

Examples:

```
        :ifxreport:gl_post::default::
        :if:test "$mz_status" = "1":exit:
```

The `:if:` Instruction

The `:if:` instruction conditionally executes any other VM instruction if the UNIX command returns a "true" (0) status. Any return status other than 0 skips the `:if:` instruction parameter.

Format:

```
      :if:UNIX_command:other_FourthGen_Menus_instruction:
```

Example:

:if:test “$mz_status” = 1:post:gl.chart:totals:a:Posting Totals:

The `:show:` Instruction

This instruction is used to announce to the user what menu item has been selected and what it does. The :show: instruction also allows the user a chance to verify his menu selection. It is especially important to include item instructions prior to instructions that print or post data.

Format:

:show:text:

The `:pause:` Instruction

The :pause: instruction is used to pause during processing until the user clicks [OK] or [Cancel] to continue, and to allow the user to define or change the destination of any program output.

Format:

```
:pause:[x|p]:[e|E]:
```

If the x flag is used, the user is given the option of exiting the current set of Instructions.

If the p flag is used, the user is able to exit, redirect output, or change the number of copies to print.

The 'e' flag means No edit option allowed (ignore $mn_modify).

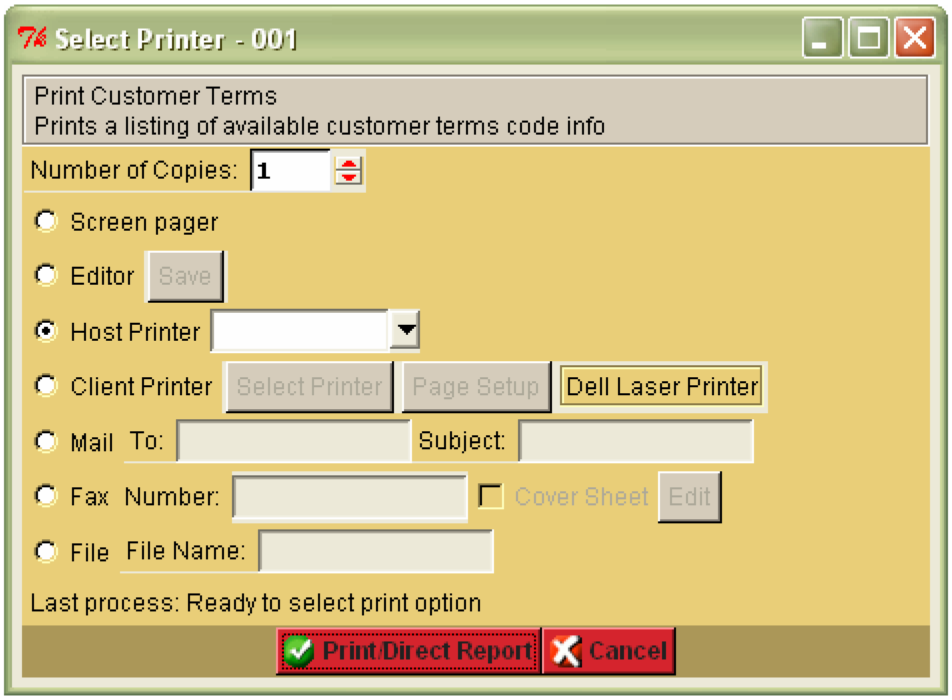
The 'E' flag means Allow edit option allowed (ignore $mn_modify).

The :pause: instruction is commonly used after :show: instructions to give the user the opportunity to confirm his selection of the menu item.

Example:

```
:show:Print Summary by Name:
:show:Prints a list of customers:
:show:Sorted by customer name:
:pause:p:
:ifxreport:ar:o_termls::default::
```

Below is the ‘Select a Printer’ Dialog box that the ‘pause:p’ instruction produces:



Print Customer Terms
Prints a listing of available customer terms code info

Number of Copies: 1

( ) Screen pager
( ) Editor Save
(●) Host Printer
( ) Client Printer Select Printer Page Setup Dell Laser Printer
( ) Mail To: Subject:
( ) Fax Number: ☐ Cover Sheet Edit
( ) File File Name:

Last process: Ready to select print option

Figure 3 - 17

At the top are the ‘:show:’ instructions that proceed the ‘:pause:p:’ instruction.

Number of Copies

Next the user can either key in the number of copies desired or can use the selector arrows to set the number copies. A series of radio buttons on the left control the selection of the output ‘media’.

Screen Pager

The option presents the report to the screen via a screen pager (See ‘report_pager’ environment variable).

Editor

Host Printer

Key in or select from a drop down box of available printers (See environment variables: ‘LPDEST’, ‘printerlist’, ‘filedir’).

Client Printer

The buttons ‘Select Printer’ and ‘Printer Setup’ execute standard Windows Printer Dialog Boxes. (See environment variable: mn_nolocalprinter) (Note the command line switch –l)

Mail

Before the mail feature can be used, you must have email configured in your computer. See the Visual Development Tools documentation for more information. Enter the name of the recipient and the subject line of the message. (See environment variables: mz_nomail, MAILER) (Note the command line switch –m)

Fax

Before the FAX feature can be used, a fax application must be installed. Enter the fax number. (See environment variables: mn_nofax, faxnum, faxcover) (Note the command line switch –f)

File

Enter a valid file name (no path). (See environment variables: filedir)

The :system: Instruction

The :system: instruction is used to execute a one-line operating system command, a program, or a shell script. If the command, program, or script to be executed cannot be found on the user's regular command path, then it must include the entire path name, just as if it were run outside of Fourth Generation Menus. If the command, program, script, or batch file fails to execute correctly, you can return the user to the menu without continuing with the rest of the instructions in the set of Item Instructions. You can use the :system: instruction to change your current working directory while in a menu.

Format:

```
:system:system_command_1:[system_command_2:]...
```

Example:

```
:system:ls -la:
```

Translated instructions:

'mz' Instructions	**Visual Menu Instructions**
`:addmenu:`	`:menu:`
`:allow:`	Security tables
`:changemenu:`	`:menu:`
`:deny:`	Security tables
`:item:`	`:show:`
`:submenu:`	`:menu:`
`:unix:`	`:system:`

Unsupported instructions:

```
:form:
:input:
:log:
:needs:
:password:
:pc:
:print:
:replace:
:skip:
:cursor:
:default:
:end:
:help:
:protect:
:swap:
:tab:
:valid:
```

Version Control

Version control allows you to run a particular version of a program. It allows you to take advantage of Version Control, which is a feature in Fourth Generation Screen that allows programmers to have different versions of a program within the same directory structure without multiple copies of files.

For example, you have several users all using the same program, but ABC user and XYZ user both need slight variations. With multi-version control, Menus allows you to run any one of those versions by setting a new environment variable called $cust_key that corresponds to the extension on the directory and files for the particular version, e.g., .abc, .xyz, .4gc, or .4gs. What Menus will do is run the program version connected with that particular $cust_key. If a particular version of the program doesn't exist for the $cust_key, it will search the program directories for the program to run in a hierarchical fashion starting with customer-specific (.abc or .xyz), to custom (.4gc), and then to base (.4gs). In the example below, we have a general ledger (gl.4gm) application with its different program directories. There are four different versions of the application that may be executed:

```
          |-- i_chart.abc
          |-- i_chart.4gc
          |-- i_chart.4gs
          |
          |
          |-- o_income.4gc
gl.4gm --|-- o_income.4gs
          |
          |
          |-- p_genled.xyz
          |-- p_genled.4gs
          |
          |
          |-- i_genjrn.4gs
```

Depending on what the $cust_key is set to, Menus will execute the different versions of the program as shown below.

```
 cust_key=4gs      i_chart.4gs
                   o_income.4gs
                   p_genled.4gs
```

```
                i_genjrn.4gs

cust_key=4gc    i_chart.4gc
                o_income.4gc
                p_genled.4gs   - since there is no p_genled.4gc
                i_genjrn.4gs   - since there is no i_genjrn.4gc

cust_key=xyz    i_chart.4gc    -  since there is no i_genjrn.xyz
                o_income.4gc   - since there is no i_genjrn.xyz
                p_genled.xyz
                i_genjrn.4gs   - since there is no i_genjrn.xyz
                                 or i_genjrn.4gc

cust_key=abc    i_chart.abc
                o_income.4gc   - since there is no o_income.abc
                p_genled.4gs   - since there is no p_genled.abc
                                  or p_genled.4gc
                i_genjrn.4gs   - since there is no i_genjrn.abc
                                 or i_genjrn.4gc
```

At any time, you can set the $cust_key variable, run Menus, and see what the product looks like for any specific user (abc or xyz), or see your value-added product (4gc).

The sequence of directories in which an application looks for a specified program is as follows:

- If there is a $cust_key variable and a program exists in the directory with the cust_key as the extension, then it runs that program.
- If there is no directory with the $cust_key extension, it looks for a custom directory with the .4gc extension and runs the program in that directory if it exists.
- If it still has not found a program to execute, it then goes into the 4GL source directory with the .4gs extension and runs the program in that directory.

This sequence allows you to maintain the standard software and any number of specific modifications in the same directory structure.

Version Control and $DBPATH

When using version control, directories listed in your $cust_path are automatically prepended to your $DBPATH. This allows the program to locate .42f files that may not appear in your local directory.

If your $cust_path is "4gc:4gs" and the directory that the program is running in is the .4gc (custom) directory, then the $DBPATH variable is prepended with "../{prog_name}.4gs:".

If the directory that the program is running in is the $cust_key (customer specific) directory, then the $DBPATH variable is prepended with "../{prog_name}.4gc:../{prog_name}.4gs:," where {prog_name} is the name of the program that is being run.

This allows you to keep form (.42f) files that are the same throughout different versions in their original location and avoid copying these files. At runtime, the form file will be found even if it isn’t in the custom directory because the $DBPATH variable also acts as a form path for GENERO BDL.

4 User’s Guide

- Menus Screens
- Menu Hierarchy
- Screen Configuration

Menus Screens

The Menu screen shown below is composed of:

- Title bar
- Menu bar
- Button bar
- Client Area-- Menus
- Status bar

See Figure 4 –01 for a sample screen layout.

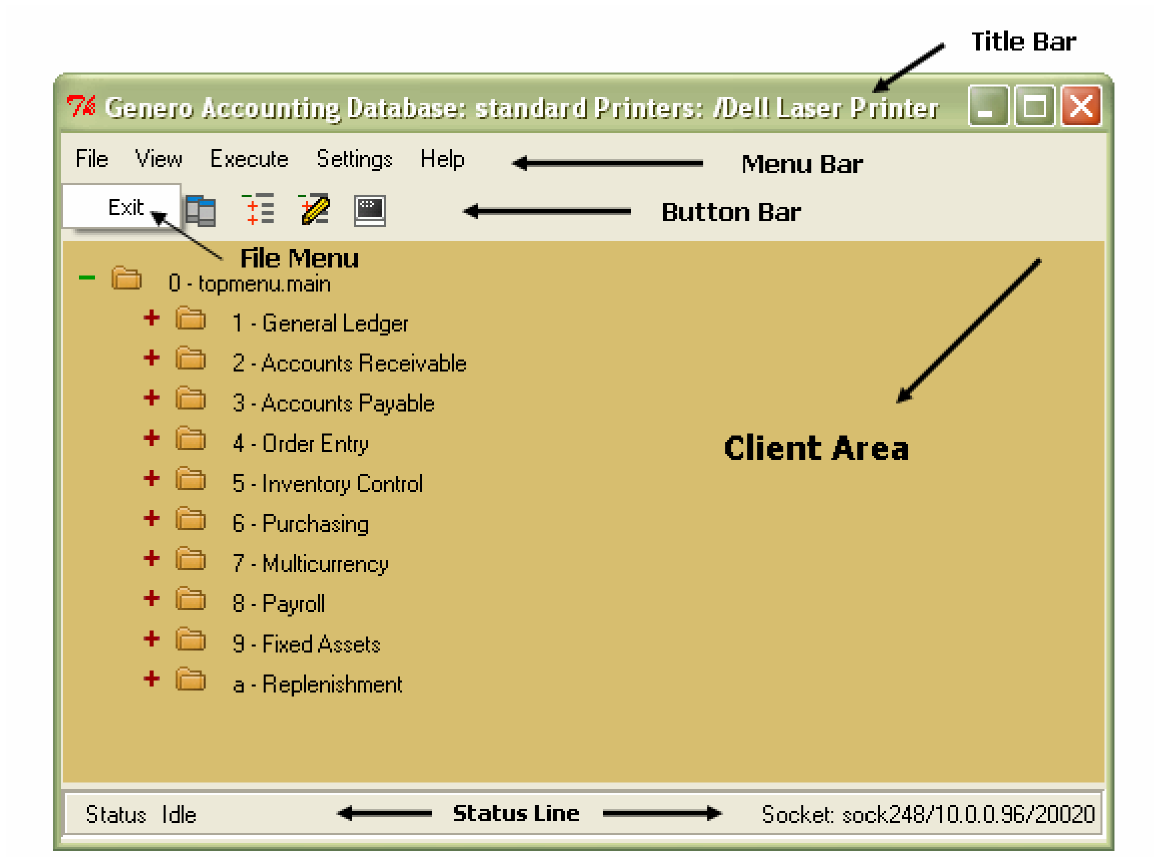


Figure 4 - 01

Menu Bar

The Menu Bar has drop-down menus that display when users click the desired menu buttons. The menu buttons, File, View, Execute, Settings, and Help are profiled here.

File Menu

The only option under the File button is to exit the program.

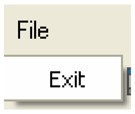


Figure 4 - 02

View Menu

You can view the accounting options in Classic mode that offers a title, or you can view the options in Explorer mode. Explorer mode gives much more detailed information such as the date and time that the file was last saved. Edit Mode is for an administrator to edit the menu structure itself.

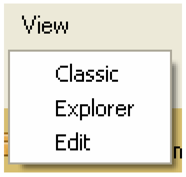


Figure 4 - 03

Execute Menu

You can launch the VM **Configuration** program. You can **Shell** out of Menus in order to work at the UNIX prompt. As an Administrator you can enter **Security** and execute any of the security programs. It is important to authorize access or restrict access to some information.

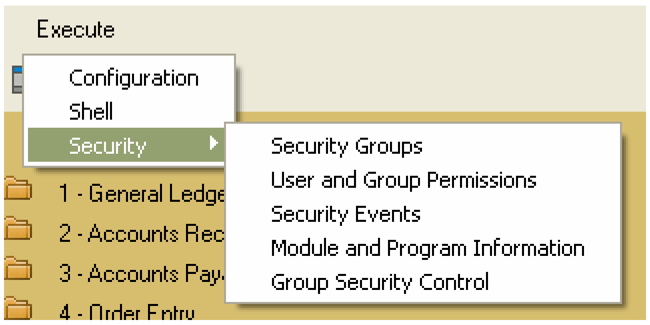


Figure 4 – 04

Settings Menu

You can also your Color/Font Configuration with **Select Resources** and can **Change database access** here.

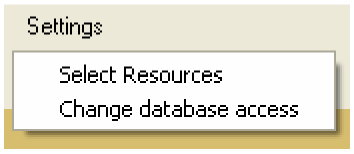


Figure 4-05

Help Menu

Offers information **About** Fourth Generation Menus.

Figure 4 - 06

Button Bar

The Button Bar is a supplement to the dropdown menus. Users can change the current View by clicking the Explorer or Classic button. Users can also Shell Out using the button bar, if they have permission to do so. (See Figure 4 – 07).

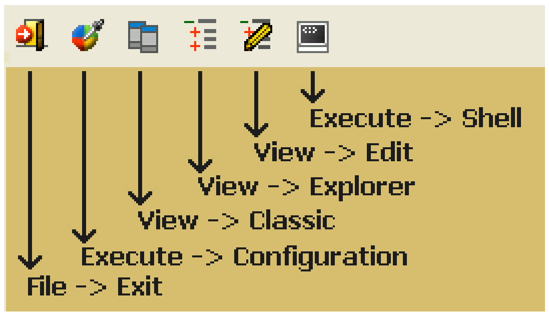


Figure 4 – 07

View Buttons

The Explorer button and the Classic button allow Users to easily change views. Notice the different between the views in Figure 4 – 08, the Explorer View, and Figure 4 – 09, the Classic View:

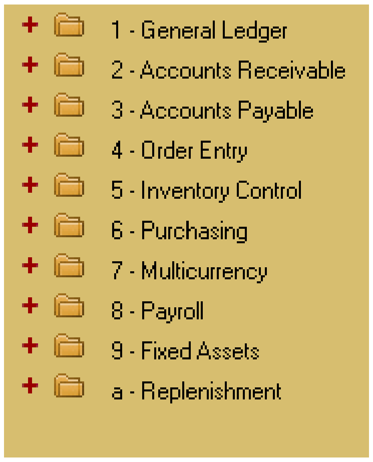


Figure 4 - 08

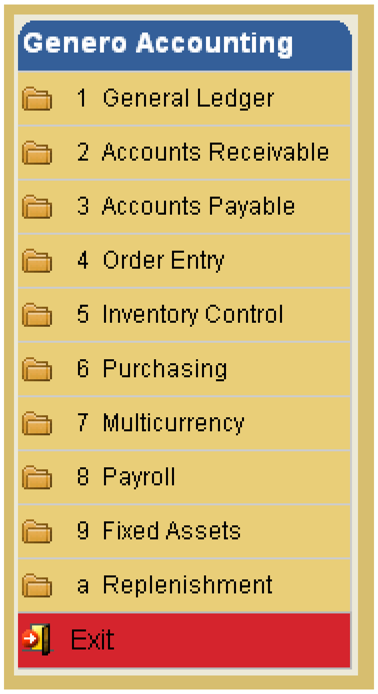


Figure 4 – 09

Menu Hierarchy

The menu ‘mainmenu.main’ is displayed in Figure 4 -11:

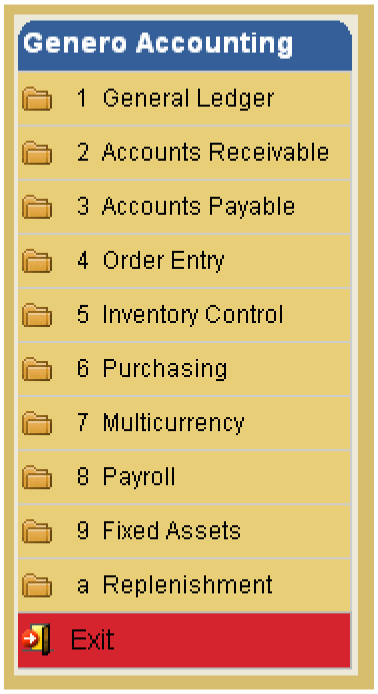


Figure 4 - 11

The folder icons on the left mean that each item is a menu (folder) that will link to another menu. Clicking the ‘Accounts Receivable’ and then on ‘Customer Information’ displays Figure 4 -11:

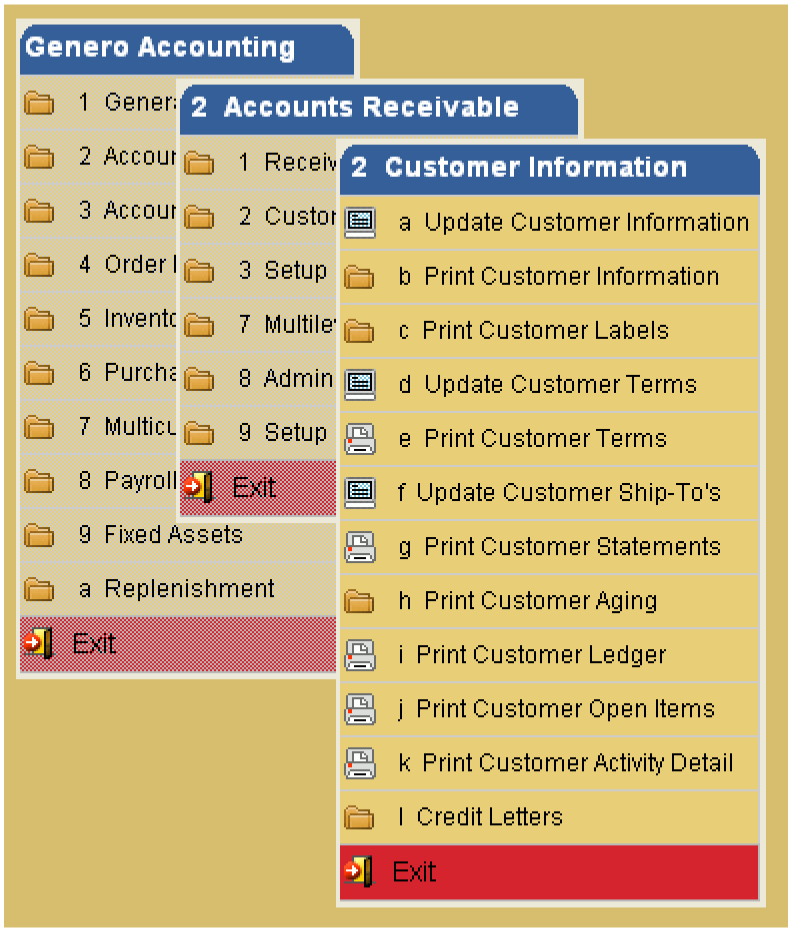


Figure 4 – 12

There is no limit to the number of Menu levels that can be established. See your System Administrator if another level needs to be added to your VM system.

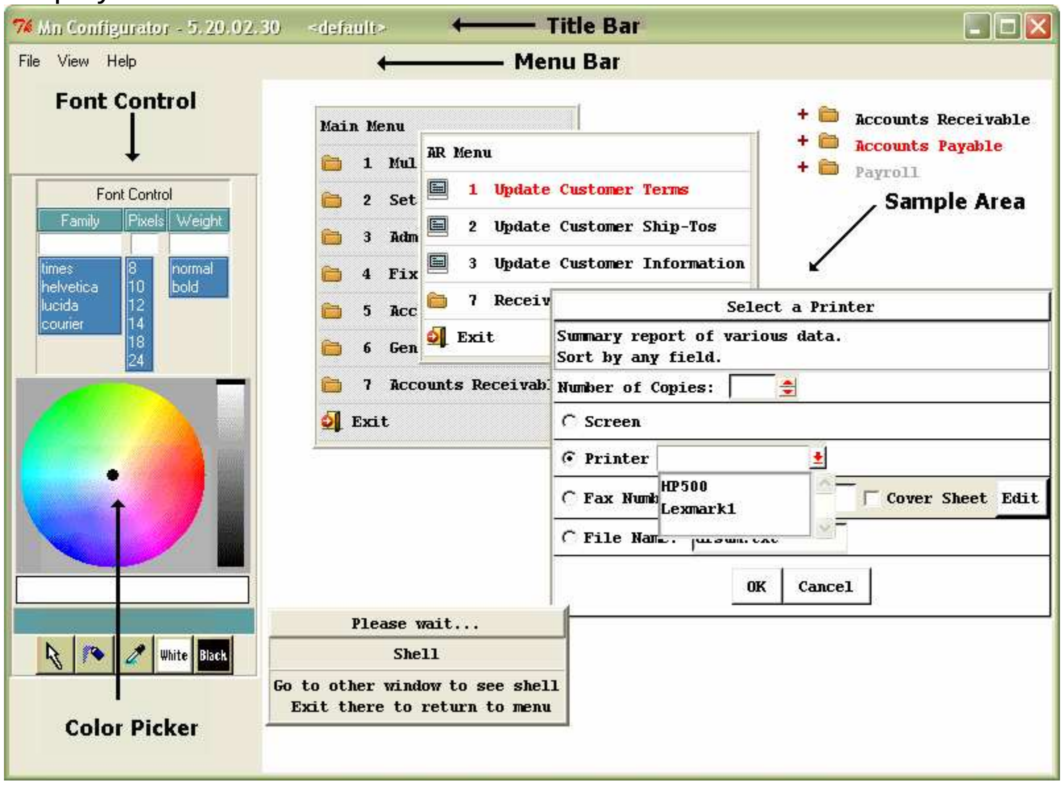
Screen Configuration

Each workstation PC can be individually configured in terms of colors and fonts by maintaining a resource file called: **<WTK>/fgss_bin/mn.res**. This file is the file opened by the 'mntk.exe' client menu program. If your company wants a uniform look and feel to the Menus product, set the color schemes as you choose, then copy the Master File **mn.res** to all workstations.

Note: There may be a number of resource files (all ending with the '.res' extension) on the workstation—for example: 'brown.res', 'plum.res', 'sepia.res'. From the menu system, any of these may be selected and will be copied to 'mn.res'.

The Main Screen

Upon executing the configuration program: , the program's main screen displays:



Summary report of various data.
Sort by any field.

Go to other window to see shell
Exit there to return to menu

Figure 4 - 12

At the top is the Title Bar, and underneath is the Menu Bar (File-View-Help). On the right side of the screen are the controls for Font and Color. At the right is a sample View of the Menus.

Menu Bar

File Menu

The File dropdown menu allows Users to create a New configuration, Open an existing configuration, Save the configuration, Save As a different file name, Delete the current configuration, or Exit the Screen Configuration utility.

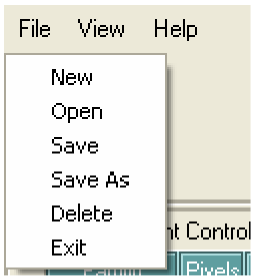


Figure 4 - 13

View Menu

The View pulldown menu allows users to toggle between Colors/Fonts editing and Place Logo editing.

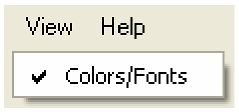


Figure 4 - 14

Help Menu

Offers information About Fourth Generation Menus.

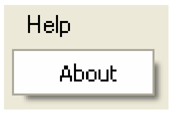


Figure 5 - 15

Sample Area

Click on **File** and select **Open**, next use the drop-down box and open ‘slate’.

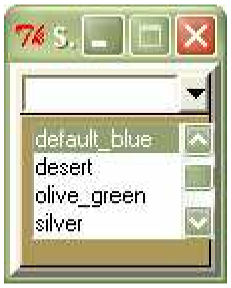


Figure 4 - 16

A sample of the 'slate' color configuration displays using the main configuration screen. Notice how the colors have changed in the Sample Configuration Display Area.

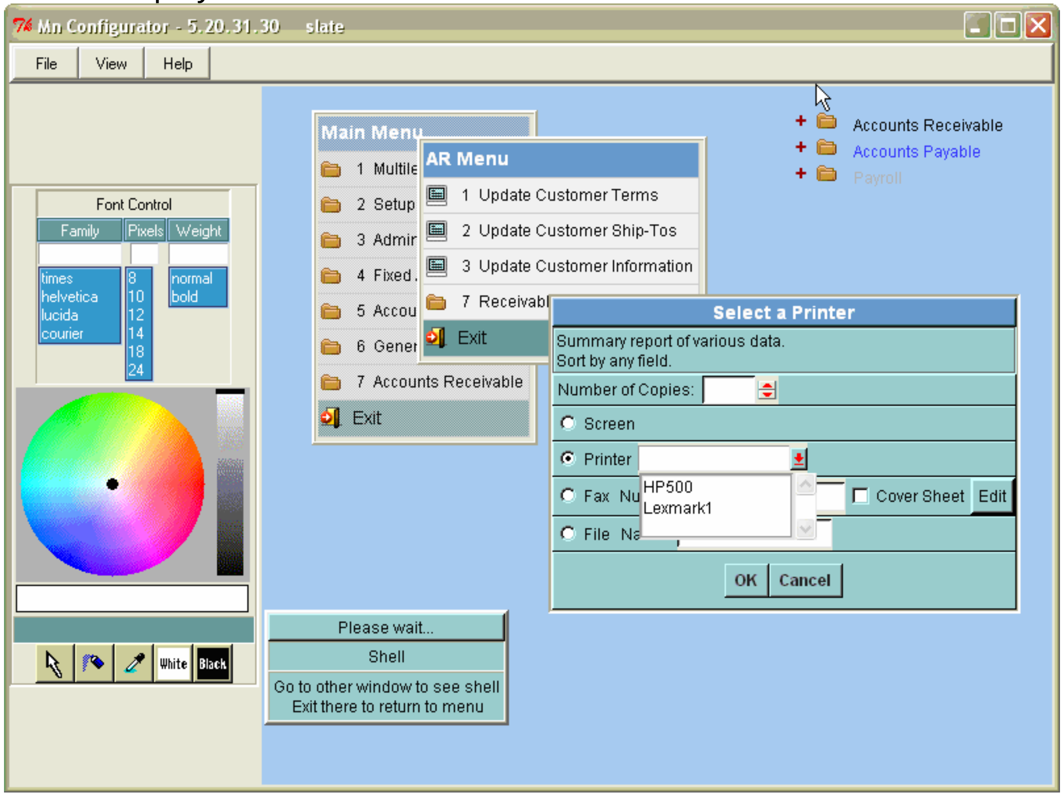


Figure 4 - 17

The Sample Area above is composed of discrete areas (widgets) that can be individually configured as to background color and foreground (text) color and font. Not all widgets have text.

To configure a widget, begin by right clicking on it and selecting from a pop-up menu which characteristic to configure:

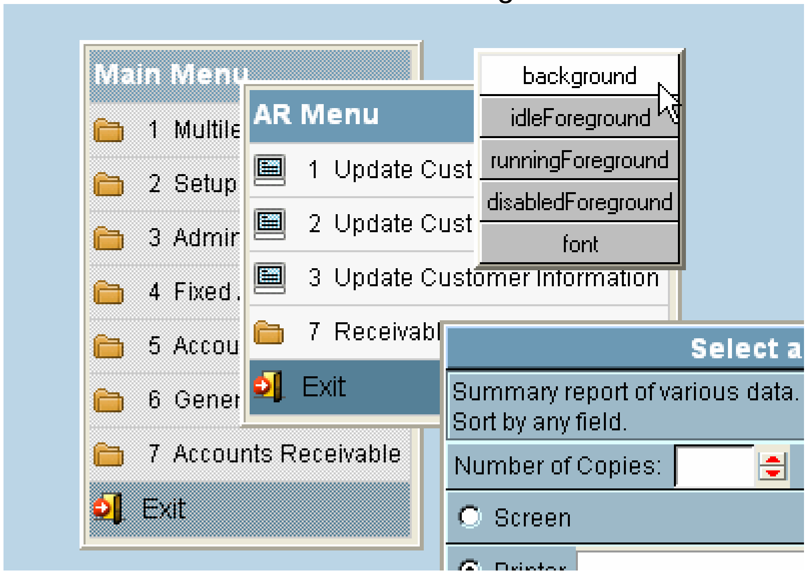


Figure 4 - 18

The user has clicked on the canvas area and has selected ‘background’.

Color Configuration

Next the current color will be filled into the ‘Current Color Rectangle’. See the close up below:

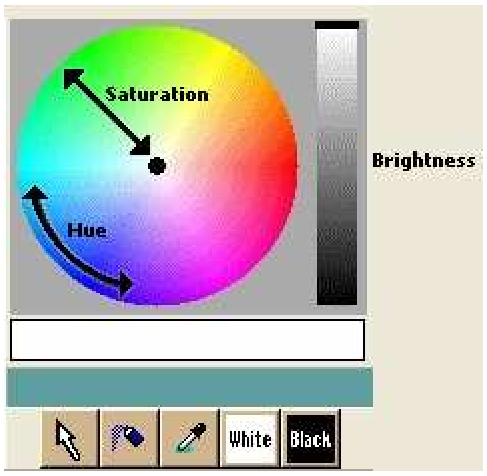


Figure 4 - 19

To change the color the user must manipulate the controls labeled above.

- Hue: Position around the perimeter of the color circle.
- Saturation: Distance from the center of the color circle.
- Brightness: Vertical distance within the right hand rectangle
- Dot: User left clicks and moves the dot to select hue and saturation.
- Bar: User left clicks and moves the bar to select brightness.
- Current Color: When the ‘Dot’ or ‘Bar’ are released the color selected is represented here and the background or foreground color of the widget is changed.

Not labeled are the buttons below the ‘Current Color’ rectangle. They are:

- Pointer: Normal mode for selecting a widget within the ‘Sample Area’
- Spray can: Next time user clicks on a widget within the ‘Sample Area’ the ‘Current Color’ will be filled in.
- Eye Dropper: Next time user clicks on a widget within the ‘Sample Area’ the background color will become the ‘Current Color’.
- White: selected (background or foreground color) will be made white.
- Black: selected (background or foreground color) will be made black.

Font Configuration

If the user selects ‘Font’ from the widget’s pop-up menu the ‘Font Control’ area is used to change the font. Study the close-up below.

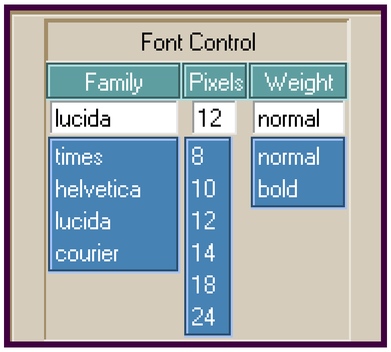


Figure 4 - 20

First, notice that the current font of the widget has been filled in. Double clicking on a ‘Family’, ‘Pixels’, or ‘Weight’ selection will change the text appropriately.

Appendix A

Change to upper level libraries

```
# cd $fglibdir/lib/report.4gs
# vi ./exitnrow.4gl
```

```
#######################################################
###############
function exit_nrows()
#######################################################
###############
# stub function called by flow_control.  if a program
must do
# something when no rows are selected for a report,
they can create
# this function locally.
#
```

# fg.make –mn –a -R

Now go into each ‘report’ directory and…
# fg.make -l

Appendix B

Menu Tables

There are two main tables plus an additional table that concerns setting of permissions. The main tables are:

- 'cgsmnitm' which contains the text that appears next to each item or option
- 'cgsmncmd' which contains the Menu Instructions (Control Language) associated with the menu's option.

The key for the'cgsmnitm' table is the field '**opt**' (a one character option—a-w, 0-9) plus a '**mname**' field. The menu item's option plus name form the main key. Generally, it would be a good idea to use names that are meaningful and hierarchical in nature. There is an additional field that marks each menu selection as either another folder or a terminal option (field '**mtype**'). A menu structure for an accounting system might have options, names, and mtypes like:

Op	Name	Mtype	
a	armenu.customer	FL	{'FL' = folder (menu)}
b	armenu.customer	SC	{'SC' = Screen maintenance}
c	armenu.customer	RP	{'RP' = Report}

Populate the menu tables.

First make sure the database engine is running, then enter the following commands:

For example:

$ company=standard ; export company (or whatever database is being loaded)
$ DBDELIMITER="~" ;export DBDELIMITER ("~" is generally a good choice, default is "|", must not conflict with any text in the flat files)
$ mz=$fg/accounting/menu ;export mz (or wherever the flat file menu structure root is)
$ cd $mz
$ $fg/bin/menucvt.sh mainmenu (or whatever is the directory of the top of the menu structure).

Look at file: "$mz/mncv.log" for errors.

Upgrade from a previous version:

To update table schemas:
It may be necessary to update the Informix table schemas (log file shows error 'Bad schema').

```
     $ $fg/bin/menucvt.sh -u mainmenu
```

To reload multilingual messages:

```
     $ $fg/bin/menucvt.sh -m:$fg/FG_tmp mainmenu
```

To load/reload security from flat files (.prm):

```
     $ $fg/bin/menucvt.sh -s mainmenu
```

Where 'mainmenu' is the top of the menu structure.

Appendix C

Menu Command Line

To use a VM menu, you must run the VM runtime program. The command to execute from the UNIX operating system prompt is:
mn [-v] [-h] [-m] [-f] [-l] [-r] [-x **option.menu_name**] [-d **directory**]

-v	Print version information and exit.
-h	Print help on command line options and exit.
-m	Suppress mail option for printer dialog output. Can also use Environment variable: ‘mz_nomail’.
-f	Suppress fax option for printer dialog output. Can also use Environment variable: ‘mn_nofax’.
-l	Suppress local printer option for printer dialog output. Can also use environment variable: ‘mn_nolocalprinter’.
-r	This flag prevents the user from accessing the UNIX shell or any shell command.
-x option.name	The -x option.menu_name This option allows you to select a specific option and menu combination to run with showing the main screen.
-d directory	By default, VM automatically uses the key: opt=’0’, mname=’topmenu.main’ to lookup the command to use as a starting folder. The -d flag is used to override this default setting. For example: -d armenu_main

Appendix D

Sample Menu Structure

For the example below, the table ‘cgsmnitm’ can be viewed as a header table and table ‘cgsmncmd’ is detail. joined by the fields ‘opt’ and ‘mname’. Fields ‘mtype’ and ‘txt’ are header information while ‘cmd’ is detail (indented).

The following VM application structure is provided as an example:

```
mname           opt  mtype txt                   cmd_________
mainmenu.main     1  FL  "General Ledger"        :menu:glmenu.main:
mainmenu.main     2  FL  "Accounts Receivable"   :menu:armenu.main:
mainmenu.main     3  FL  "Accounts Payable"      :menu:apmenu.main:
mainmenu.main     4  FL  "Order Entry"           :menu:oemenu.main:
mainmenu.main     5  FL  "Inventory Control"     :menu:icmenu.main:

armenu.main       1  FL  "Receivable Ledger"     :menu:armenu.ledger:
armenu.main       2  FL  "Customer Information"  :menu:armenu.customer:
armenu.main       3  FL  "Setup Receivables"     :menu:armenu.arsetup:

armenu.customer  a  SC  "Customer Information"  :ifxscreen:ar:i_custr:::
armenu.customer  b  FL  "Print Customer Info"   :menu:armenu.custinfo:
armenu.customer  c  FL  "Print Customer Labels" :menu:armenu.label:
armenu.customer  d  SC  "Update Customer Terms" :ifxscreen:ar:i_termr:::
armenu.customer  e  RP  “Print Customer Terms"  :show:Print Customer Terms:
                                                :show:List of available customer terms:
                                                :pause:p:
                                                :ifxreport:ar:o_termls::default:::
armenu.customer  f  SC  "Update Cust Ship-To's" :ifxscreen:ar:i_shipr:::
```

Appendix E

Miscellaneous Notes

Error "Form file not found". For example:

A browse form cannot be resolved.

```
$ cust_path=css:4gs
```

and form:

`$fg/accounting/oe.4gm/i_custr.4gs/browse.per` exists but
`$fg/accounting/oe.4gm/i_custr.css/browse.per` does NOT.

Solution: Copy "browse.per" and/or "browse.42f" into i_custr.css.

Indexes:

```
i_cgsmnitm1         informix  unique  No        ckey
                                                mname
                                                opt

i_cgsmnitm2         informix  dupls   No        mname

i_cgsmnitm3         informix  dupls   No        ckey
                                                mname

CREATE INDEX i_cgsmnitm2 ON cgsmnitm(mname);
CREATE INDEX i_cgsmnitm3 ON cgsmnitm(ckey,mname);
==========================================================
i_cgsmncmd1         informix  unique  No        ckey
                                                mname
                                                opt
                                                ilevel

i_cgsmncmd2         informix  dupls   No        ckey
                                                mname
                                                opt
==========================================================
i_cgsmnsec1         informix  unique  No        ckey
                                                mname
                                                opt
                                                either_id

i_cgsmnsec2         informix  dupls   No        ckey
                                                mname
                                                opt
```

find-zombie.sh

```
#!/bin/sh
#
# find-zombie.sh
#
ps -eo pid,user,cmd | grep " mn " | grep -v "grep" | \
  sed 's/^ *//g' | cut -f1 -d\  | while read p ;do
  if ps -eo cmd | grep "mnc -p:${p} " | grep -v "grep" > /dev/null 2>&1 ;then
    :
  else
    ps --pid ${p} -h -o pid,user,cmd
  fi
done
#
# End...
```